Domain- and Quality-aware Requirements Engineering for Law-compliant Systems

Problem- and Pattern-Based Requirements Elicitation, Analysis, Reconciliation, and Optimization for Heterogeneous Needs and Stakeholders

Der Fakultät für Ingenieurwissenschaften,
Abteilung Informatik und Angewandte Kognitionswissenschaft
Universität Duisburg-Essen

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften (Doktor-Ingenieur - Dr.-Ing.)

vorgelegte und genehmigte Dissertation

von

Stephan Faßbender
aus
Bonn / Bad Godesberg

Datum der Einreichung:
14.01.2016
Datum der mündlichen Verteidigung:
11.05.2016

Erstgutachter: Prof. Maritta Heisel
Zweitgutachter: Prof. Ketil Stølen

Contents

List of Figures

List of Tables

List of Acronyms

A	Assumption or Attidute
ACM	Association for Computing Machinery
AHP	Analytic Hierarchy Process
AMI	Advanced Metering Infrastructure
ANP	Analytic Network Process
ARIS	Architektur Integrierter Informationssysteme (Architecture of Integrated InformationSystems)
BABOK	Business Analysis Body Of Knowledge
BDSG	Bundes DatenSchutz Gesetz (federal dataprotection act)
BOCR	Benefits Opportunities Costs Risks
BPMN	Business Process Modeling Notation
CC	Common Criteria
CIA	Confidentiality Integrity Availability
CMMi	Capability Maturity Model Integration
CORE	Computing Research & Education
CPU	Central Processing Unit
CRM	Customer Relationship Management
CSAP	Cloud System Analysis Pattern
CSP	Creative Problem Solving
D	Domainknowledge
DD	Definition Designation
DK	Domain Knowledge
DSL	Domain Specific Language
EBNF	Extended Backus-Naur Form
ECL	Epsilon Comparison Language
EGL	Epsilon Generation Language
EMF	Eclipse Modeling Framework
EOL	Epsilon Object Language
EPC	Electronic Process Chain
EPL	Epsilon Pattern Language
ERA	Excellence in Research for Australia
ERD	Entity Relationship Diagram
ETL	Epsilon Transformation Language
EU	European Union
EVL	Epsilon Validation Language
EWL	Epsilon Wizard Language
F	Fact
FAD	Functional Analysis Diagram
FR	Functional Requirement
G	Goal
GEMA	Gesellschaft für musikalische Aufführungs- und mechanische Vervielfältigungsrechte
GG	GrundGesetz (basic law)

GMF	Graphical Modeling Framework
HAN	Home Area Network
HIPAA	US Health Insurance Portability and Accountability Act
ICT	Information and Communication Technology
IDE	Integrated Development Environment
IDEF	Integrated DEFinition methods
IEC	International Electrotechnical Commission
ISO	International Standardization Organization
IT	Information Technology
LadSchlG	Ladenschlußgesetz (store-closing Act)
LAN	Local Area Network
LIP	Law (Identification) Pattern
LMN	Local Metrological Network
MILP	Mixed Integer Linear Programming
MOF	Meta Object Facility
MOO	Multi Objective Optimization
MRQ	Main Research Question
MVC	Model View Controller
NFR	Non-Functional Requirement
NLP	Natural Language Processing
NSA	National Security Agency
OCL	Object Constrain Language
OED	Oxford English Dictionary
OMG	Object Management Group
OS	Operating System
P	Plan or Process
P2P	Peer to Peer
PAIS	Process Aware Information System
PFI	Problem Frame Identification
PICOC	Population Intervention Comparison Outcomes Context
PresSuRE	Problem-based Security Requirements Elicitation
ProPAn	Problem-Based Privacy Analysis
Q	Quality
QR	Quality Requirement
R	Requirement
RAM	Random Access Memory
RCP	Rich Client Platform
RE	Requirements Engineering
RQ	Research Question
RUP	Rational Unified Process
SAT	Satisfiability
SD	Strategic Dependencies
SDLC	Secure Development Life Cycle
SE	Software Engineering
SLA	Service Level Agreement
SLR	Structured Literature Review
SOA	Service Oriented Architecture
SOALSP	SOA (Layer / Stakeholder) Pattern
SOAML	SOA Modeling Language
SR	Strategic Rational

SR	Security Requirement
SWEBOK	SoftWare Engineering Body Of Knowledge
SWT	Standard widget Toolkit
SysML	Sysms Modeling Language
TC	Transformation Card
TMG	TeleMedien Gesetz (tele-media act)
TOE	Target Of Evaluation
UK	United Kingdom
UML	Unified Modeling Language
UML4PF	Unified Modeling Language for (4) Problem Frames
UP	Unified Process
UPROM2PF	UPROM to (2) Problem Frames
US	United States
USA	United States of America
VG Wort	Verwertungsgesellschaft Wort
WAN	Wide Area Network
XMI	Xml Metadata Exchange

Abstract

The long known credo of requirements engineering states that it is challenging to build the right system if you do not know what right is. There is strong evidence that this credo exactly defines and describes the necessity of requirements engineering. Fixing a defect when it is already fielded is reported to be up to eighty times more expensive than fixing the corresponding requirements defects early on. In general, conducting sufficient requirements engineering has shown to be a crucial success factor for software development projects. Throughout the progression from initial stakeholders' wishes regarding the system-to-be to a specification for the system-to-be requirements engineers have to undergo a complex decision process for forming the actual plan connecting stakeholder wishes and the final specification. Indeed, decision making is considered to be an inherent part of requirements engineering. In this thesis, we try to understand which activities and information are needed for selecting requirements, which the challenges are, how an ideal solution for selecting requirements would look like, and where the current state of the art is deficient regarding the ideal solution.

Within this thesis we identify the information necessary for an informed requirements selection, present a process in which one collects all the necessary information, highlight the challenges to be addressed by this process and its activities, and a selection of methods to conduct the activities of the process. All the collected information is then used for an automated requirements selection using an optimization model which is also part of the contribution of this thesis. As we identified two major gaps in the state of the art considering the proposed process and its activities, we also present two novel methods for context elicitation and for legal compliance requirements elicitation to fill the gaps as part of the main contribution.

Our solution for context elicitation enables a domain-specific context establishment based on patterns for different domains. The context patterns allow a structured elicitation and documentation of relevant stakeholders and technical entities for a system-to-be. Both, the documentation in means of graphical pattern instances and textual template instances as well as the method for collecting the necessary information are explicitly given in each context pattern. Additionally, we also provide the means which are necessary to derive new context patterns and extend our context patterns language which is part of this thesis.

Our solution for legal compliance requirements elicitation is a pattern-based and guided method which lets one identify the relevant laws for a system-to-be, which is described in means of functional requirements, and which intertwines the functional requirements with the according legal requirements. This method relies on the collaboration of requirements engineers and legal experts, and bridges the gap between their distinct worlds.

Our process is exemplified using a running example in the domain of service oriented architectures. Additionally, the results of applying (parts of) the process to real life cases from the smart grid domain and voting system domain are presented, as well as all other results from the scientific means we took to ground and validate the proposed solutions.

Abstract (German)

Der bekannte Leitsatz in der Anforderungserhebung und -analyse besagt, dass es schwierig ist, das richtige System zu bauen, wenn man nicht weiß, was das "Richtige" eigentlich ist. Es existieren überzeugende Belege, dass dieser Leitsatz die Notwendigkeit der Anforderungserhebung und -analyse exakt definiert und beschreibt. Zum Beispiel ergaben Studien, dass das Beheben von Defekten in einer Software, die bereits produktiv genutzt wird, bis zu 80 mal so teuer ist wie das frühzeitige Beheben der korrespondierenden Defekte in den Anforderungen. Generell hat es sich gezeigt, dass das Durchführen einer angemessenen Anforderungserhebung und -analyse ein wichtiger Erfolgsfaktor für Softwareentwicklungsprojekte ist. Während der Progression von den initialen Wünschen der beteiligten Interessensvertretern für ein zu entwickelndes System zu einer Spezifikation für eben dieses Systems müssen Anforderungsanalysten einen komplexen Entscheidungsprozess durchlaufen, der die initialen Wünsche in die Spezifikation überführt. Tatsächlich wird das Treffen von Entscheidungen als integraler Bestandteil der Anforderungsanalyse gesehen. In dieser Arbeit werden wir versuchen zu verstehen welche Aktivitäten und Information von Nöten sind, um eine fundierte Auswahl von Anforderungen vorzunehmen, welche Herausforderungen damit verbunden sind, wie eine ideale Lösung zur Anforderungswahl aussehen könnte und in welchen Bereichen der aktuelle Stand der Technik in Bezug auf diese ideale Lösung lückenhaft ist.

Innerhalb dieser Arbeit werden wir die Informationen, die notwendig für eine fundierte Anforderungsauswahl sind, identifizieren, einen Prozess präsentieren, um diese notwendigen Informationen zu sammeln, die Herausforderungen herausstellen, die durch diesen Prozess und die damit verbundenen Aktivitäten adressiert werden und eine Auswahl von Methoden diskutieren, mit deren Hilfe man die Aktivitäten des Prozesses umsetzen kann. Die gesammelten Informationen werden dann für eine automatisierte Anforderungsauswahl verwendet. Für die Auswahl kommt ein Optimierungsmodell, das Teil des Beitrags dieser Arbeit ist, zum Einsatz. Da wir während der Erstellung dieser Arbeit zwei große Lücken im Stand der Technik bezüglich unseres Prozesses und der damit verbundenen Aktivitäten identifiziert haben, präsentieren wir darüber hinaus zwei neuartige Methoden für die Kontexterhebung und die Erhebung von rechtlichen Anforderungen, um diese Lücken zu schließen. Diese Methoden sind Teil des Hauptbeitrags dieser Arbeit.

Unsere Lösung für der Erhebung des Kontext für ein zu entwickelndes System ermöglicht das Etablieren eines domänenspezifischen Kontextes unter Zuhilfenahme von Mustern für verschiedene Domänen. Diese Kontextmuster erlauben eine strukturierte Erhebung und Dokumentation aller relevanten Interessensvertreter und technischen Entitäten für ein zu entwickelndes System. Sowohl die Dokumentation in Form von grafischen Musterinstanzen und textuellen Vorlageninstanzen als auch die Methode zum Sammeln der notwendigen Informationen sind expliziter Bestandteil jedes Kontextmusters. Zusätzlich stellen wir auch Hilfsmittel für die Erstellung neuer Kontextmuster und das Erweitern der in dieser Arbeit präsentierten Kontextmustersprache zur Verfügung.

Unsere Lösung für die Erhebung von rechtlichen Anforderungen basiert auch auf Mustern und stellt eine Methode bereit, welche es einem erlaubt, die relevanten Gesetze für ein zu erstellendes System, welches in Form der funktionalen Anforderungen bereits beschrieben sein muss, zu identifizieren und welche die bestehenden funktionalen Anforderungen mit den rechtlichen Anforderungen verknüpft. Diese Methode beruht auf der Zusammenarbeit zwischen Anforderungs-

analysten und Rechtsexperten und schließt die Verständnislücke zwischen ihren verschiedenartigen Welten.

Wir veranschaulichen unseren Prozess unter der Zuhilfenahme eines durchgehenden Beispiels aus dem Bereich der service-orientierten Architekturen. Zusätzlich präsentieren wir sowohl die Ergebnisse der Anwendung unseres Prozesses (bzw. Teilen davon) auf zwei reale Fälle aus den Bereichen von Smart Grids und Wahlsystemen, als auch alle anderen Ergebnisse der wissenschaftlichen Methoden, die wir genutzt haben, um unsere Lösung zu fundieren und validieren.

Acknowledgment

A first acknowledgment goes to my scientific sparring partners including the anonymous reviewers, people i worked with in projects, people I met at conferences, and, most importantly, my colleagues. I really appreciate all the discussions, different viewpoints, new insights, tips and feedback provided to me. All the paper reviews, promi-workshops[1], FoKos[2], dinner discussions, and so forth were indispensable for improving my research and polishing the thesis as it is. Special thanks on this part goes to Kristian Beckers and Rene Meis for all the collaborative work we have done together. I owe you tons.

Not to forget my supervisor Maritta Heisel with all her valuable feedback, guidance, and the chance she gave me to actually pursue the aim of writing this thesis. I apologize that you had and have to read, review, and judge all the content written down in this thesis. The same goes to Ketil Stølen, my second supervisor.

Hats up to Christina Menges for all the proofreading she has done.

Of course, all my gratitude goes also to the crazy pack which is called my family. In the end I can not list all the things for which I would like to say "Thank you" to my siblings Ruth, Andi, and Michi and my mother Regina not only regarding the time I was writing this thesis.

A special person to me is my wife Polina. I do not find any better words for expressing my thankfulness. Hence, I make it the scientific way (short and precise): Thank you, Polina. I hope you know what you mean to me.

[1] A workshop held on regular basis for all the PhDs of our working group

[2] Forschungskolloquium (research colloquium)

Part I.

Foundations

Chapter 1

Introduction

In this chapter we introduce the matter of requirements engineering and its importance for successful software development (Section 1.1). Afterward, we discuss the relation to decision making and motivate the importance of requirements selection (Section 1.2), and derive the research question which we discuss within this thesis (Section 1.3). Based on a problem & gap study (see Chapter 3[1] for a definition) on the matter of decision making and requirements selection, we show which kind of information is important for requirements selection, which challenges one has to face when forming a solution for requirements selection, and finally propose a general process for collecting all this information and conducting the requirements selection (Section 1.4[2]). Finally, we introduce our own papers which are of relevance for this thesis and outline the structure of the whole work (Section 1.5[3]).

1.1. Context

The long known credo of requirements engineering states that it is challenging to build the right system if you do not know what right is:

> "The hardest single part of building a software system is deciding precisely what to build. ... No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later."(Brooks, 1995 [75])

There is strong evidence that this credo exactly defines and describes the necessity of requirements engineering. Fixing a defect when it is already fielded is reported to be up to eighty times more expensive than fixing the corresponding requirements defects early on (Willis, 1998 [391]; Boehm and Papaccio, 1988 [61]; Buchan, Ekadharmawan, and MacDonell, 2009 [79]). In general, conducting sufficient requirements engineering has shown to be a crucial success factor for software development projects (Sadraei, Aurum, Beydoun, and Paech, 2007 [332]; Thevenet and Salinesi, 2007 [368]; Buchan et al., 2009 [79]; Hofmann and Lehner, 2001 [185]; Ahonen and Savolainen, 2010 [2]; Monsalve, April, and Abran, 2012 [272]; Hall, Beecham, and Rainer, 2002 [169]). It has even been observed that the importance of requirements engineering is increasing over time.(Hofmann and Lehner, 2001 [185]). In consequence, an empirical study of Hall et al. (2002 [169]) in twelve software companies identifies requirements engineering as top-most problem in the development cycle which is seconded by another study of Nikula, Sajaniemi, and Kälviäinen (2000 [285]) which states the requirements engineering improvement is one of the top issues for small and medium sized companies.

While the majority of the RE problems are related to organizational issues, still 37% of the problems are related to the execution of requirements engineering itself (Hall et al., 2002 [169]).

[1] Page 41
[2] Page 9
[3] Page 22

The top four reasons for the problems are vague requirements, insufficient requirements processes, requirements growth, and complex systems. (Hall et al., 2002 [169]). While requirement processes and methods are expected to have improved in the last 10 years, the other three reasons are still a problem (vague requirements) or became even more problematic over time (number of requirements, and complexity of systems). But still, there seems to be even a need for an improvement of requirements engineering methods as many researchers state that there are many methods which cover parts of a complete requirements engineering process, but there is a need for methods integrating all those fragmentary methods (Nicolas and Toval, 2009 [283]; Cheng and Atlee, 2007 [100]; Frankova, Seguran, Gilcher, Trabelsi, Doerflinger, and Aiello, 2011 [151]; Maiden and Rugg, 1996 [250]).

Beside the improvement of requirements engineering as such, a recent development in the industry demands ways to integrate requirements engineering into more general business activities. It has been shown that alignment of IT development with business strategy leads to superior business performance (Bleistein, Cox, and Verner, 2004 [59]; Thevenet and Salinesi, 2007 [368]). In consequence, there is a need in practice for methods supporting this alignment (Thevenet and Salinesi, 2007 [368]; Singh and Woo, 2009 [355]), which means, for example, an integration of requirements engineering with established activities within organizations such as IT planning (Bleistein et al., 2004 [59]) or business process modeling (Jamshidi and Pahl, 2012 [204]; Liew, Kontogiannis, and Tong, 2004 [242]; Mayr, Kop, and Esberger, 2007 [262]; Bleistein et al., 2004 [59]; Demirörs, Gencel, and Tarhan, 2003 [114]; Aysolmaz and Demirörs, 2014 [25]; Li, van den Akker, Brinkkemper, and Diepen, 2010 [239]; Aysolmaz and Demirörs, 2014 [26]; Nuseibeh and Easterbrook, 2000 [290]). Such an integration into business activities of course adds a great magnitude of complexity, for example, to decisions taken within a requirements engineering process, and increases the importance of requirements engineering within companies once more.

But what does requirements engineering actually mean? Zave (1997 [398]) provides a good definition, which is also used by other researchers, for example, Nuseibeh and Easterbrook (2000 [290]):

> "Requirements engineering is the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behavior, and their evolution over time and across software families"

This definition covers three aspects of requirements engineering: The why, the what, and the which(Nuseibeh and Easterbrook, 2000 [290]). Why there is a need for a system-to-be, what problem shall be solved by the system-to-be, and in which way the system-to-be shall solve the problem.

Hence, a central part of requirements engineering is concerned with describing the problem that the software has to solve in a precise way (Cheng and Atlee, 2007 [100]). The problem is located in the environment in which the machine (the thing to be build) will be integrated and not in the computer (Jackson, 2001 [200]). Therefore, reasoning about the requirements involves reasoning about the environment and the assumptions made about it (Cheng and Atlee, 2007 [100]). Zave and Jackson define the three terms *requirements (**R**)*, *domain knowledge (**D**)*, and *specification (**S**)* in their extensive work (Zave and Jackson, 1997 [399])[4]. The requirements describe the desired system after the machine is built. The domain knowledge represents the relevant parts of the problem world. The specification describes the behavior of the machine in order to meet the requirements. These three descriptions are related through the entailment relationship $\mathbf{D}, \mathbf{S} \vdash \mathbf{R}$, expressing that the specification within the context of the domain knowledge should satisfy the requirements.

[4] A more precise definition of the terms used by (Zave and Jackson, 1997 [399]) is given in Section 2.2.4

This relationship of D, S and R is seen by many researchers as core ontology for describing the matter of requirements engineering (Jureta, Mylopoulos, and Faulkner, 2008 [206]). But Jureta et al. (2008 [206]) see this definition as deficient for several reasons:

> "Firstly, it does not allow partial fulfillment of (some) requirements, e.g., non-functional ones.... Secondly, the characterization leaves no room for one specification to be "better" than another... Thirdly, important notions such as those of non-functional requirements and preferences (or, "nice to have" requirements) are left out of the framework..." (Jureta et al., 2008 [206])

To fix these deficiencies, Jureta et al. (2008 [206]) propose a revised core ontology:

> "Starting from from compulsory and optional domain assumptions ($\mathbf{K}_C$ and $\mathbf{K}_O$), goals ($\mathbf{G}_C$ and $\mathbf{G}_O$), quality constraints ($\mathbf{Q}_C$ and $\mathbf{Q}_O$), softgoals ($\hat{\mathbf{Q}}_C$ and $\hat{\mathbf{Q}}_O$), Plans ($\mathbf{P}_C$ and $\mathbf{P}_O$), and attitudes ($\mathbf{A}^{>}{}_C$ and $\mathbf{A}^{>}{}_O$) communicated by the stakeholders, the requirements problem amounts to finding the plan $\mathbf{P}^*$ such that:
>
> 1. $\mathbf{K}^*, \mathbf{P}^* \mathrel{|\!\sim} \mathbf{G}^*, \mathbf{Q}^*, \mathbf{A}^{>}$
> 2. ..." (Jureta et al., 2008 [206])

This definition states that requirements engineering is about finding the plan $\mathbf{P}^*$ which fulfills the goals of the stakeholders which imply the functionality of the system-to-be ($\mathbf{G}$), quality constrains ($\mathbf{Q}$) which approximate softgoals ($\hat{\mathbf{Q}}$) of the stakeholders, and preferences ($\mathbf{A}^{>}$) of the stakeholders regarding the goals and softgoals, in the best possible way under the given domain knowledge ($\mathbf{K}$). "*" denotes here that the compulsory aspects have to be covered as far as possible[5] by the plan, but the optional parts might be neglected. A plan ($\mathbf{P}$) in terms of (Jureta et al., 2008 [206]) does not only describe the requirements of the system-to-be, but also all other measurements and actions taken, such as process definitions, user training, and so forth. A plan $\mathbf{P}$ is optimal whenever there is no other plan which addresses the preferences of all stakeholders in a better way.

Jureta et al. (2008 [206]) do not use $\vdash$ as relation, but the defeasible consequence relation $\mathrel{|\!\sim}$ which is non-monotonic in contrast to $\vdash$. They argue that $\vdash$ implies that newly learned knowledge is not allowed to invalidate $\mathbf{D}, \mathbf{S} \vdash \mathbf{R}$, which would mean that the fulfillment of $\mathbf{D}, \mathbf{S} \vdash \mathbf{R}$ has to foresee all future changes in D and that D is complete. $\mathrel{|\!\sim}$ relaxes this relation allowing a reasoning just for the current state of knowledge. Here, we have another point of view. There are two kinds of newly learned knowledge. First, knowledge which does not contradict our previous knowledge. Hence, our reasoning done to show $\vdash$ is just fine. Second, knowledge which contradicts our previous knowledge. In this case, also our reasoning might be flawed, and, of course, we have to reason about the fulfillment of $\vdash$ once again. If we now relax the relation to $\mathrel{|\!\sim}$, we would consider a reasoning to be valid which is already proven to be wrong, which is not acceptable. Hence, we would also reason once again, even when using $\mathrel{|\!\sim}$ as relation. The strong point for using $\mathrel{|\!\sim}$ is that it allows some kind of fuzzy reasoning, because we do not need a hard proof for every reasoning, but are also allowed to do a fuzzy reasoning including, for example, approximations which is unavoidable when one reasons about concepts such as, for example, softgoals.

Jureta et al. (2008 [206]) claim that their proposed ontology replaces the original one of (Zave and Jackson, 1997 [399]). In contrast, we argue that the ontology of Jureta et al. (2008 [206]) wraps the one of Zave and Jackson (1997 [399]). From our point of view, Zave and Jackson (1997 [399]) focused solely on the "what" of requirements engineering as defined by Zave (1997 [398]), and Nuseibeh and Easterbrook (2000 [290]), while Jureta et al. (2008 [206])

[5] For example, conflicting requirements cannot be covered at the same time, even when they are compulsory. Hence, here a plan has to include the decision which one to take.

have the "why" and "which" in mind. Hence, for us $\mathbf{D}, \mathbf{S} \vdash \mathbf{R}$ describes the problem which is then solved by the machine under specific circumstances described by the domain knowledge. To specify the system-to-be is the final goal of requirements engineering. Hence, the ontology (Zave and Jackson, 1997 [399]) describes the final goal. Jureta et al. (2008 [206]) focus more on the starting point of requirements engineering which is defined by the stakeholders, their goals, and preferences, and they also highlight the decision problem which is about the problems which are solved using other measurements than the system-to-be, the problems to be actually solved by the system-to-be, and to which degree the underlying goals are satisfied. But still, the requirements $\mathbf{R}$ are at least a part of the plan $\mathbf{P}^*$ ($\mathbf{R} \subseteq \mathbf{P}^*$) and in the end we are still seeking $\mathbf{S}$ for which the relation $\mathbf{D}, \mathbf{S} \vdash \mathbf{R}$ is valid. The domain knowledge $\mathbf{D}$ with respect to the requirements can be directly derived from $\mathbf{K}$, which describes the general domain knowledge, and $\mathbf{P}^* \setminus \mathbf{R}$ because all the measurements taken within the plan beside the requirements are also domain knowledge from the perspective of the system-to-be ($\mathbf{D} = \mathbf{K} \cup \mathbf{P}^* \setminus \mathbf{R}$). This way, we connect the ontology of Jureta et al. (2008 [206]) and the one of Zave and Jackson (1997 [399]), which is reasonable because in the beginning we start with the actual stakeholders, their preferences, and goals, we have to make up a plan how to satisfy the stakeholders, but in the end we are seeking a specification for our machine.

The combination of the two ontologies highlights an interesting and important aspect of requirements engineering: Throughout the progression from initial stakeholders' wishes regarding the system-to-be to a specification for the system-to-be we have to undergo a complex decision process for forming the actual plan connecting stakeholder wishes and the final specification. We investigate and discuss this finding in the next section.

1.2. Motivation for this Thesis

Indeed, decision making is considered to be an inherent part of requirements engineering (Saliu and Ruhe, 2007 [334]; Svensson, Gorschek, Regnell, Torkar, Shahrokni, Feldt, and Aurum, 2011 [365]; Aurum and Wohlin, 2003 [22]; Regnell, Paech, Aurum, Wohlin, Dutoit, and Dag, 2001 [319]):

> "Requirements can be viewed as the results of stakeholders' decisions regarding the functionality and quality of the software product to be constructed. Furthermore, the Requirements Engineering (RE) process needs staffing, planning, control, and organization; all these issues are related to decision making."(Regnell et al., 2001 [319])

Here, Regnell et al. (2001 [319]) even broaden the activities related to requirements engineering as they also include the complete planning of the development of a system-to-be. Several field studies support this view. A study of Nikula et al. (2000 [285]) showed that support for requirements selection is in the top 10 issues for small and medium size enterprises, and that the requirements selection is not only influenced by the goal to satisfy the stakeholders, but also by project management related concerns such as, for example, time, budget, and so forth. Lehtola, Kauppinen, and Kujala (2004 [234]) formulate a similar observation in their work.

In the science world, several researchers highlight the importance of requirements selection in particular and decision making within requirements engineering in general. The statements highlighting the importance range from

- focusing and improving the most important requirements

  > "Knowing the most important requirements for a software product is key to any software product improvement actions and gives increased certainty that one is building in the most important functionalities and qualities, ..."(Daneva and Herrmann, 2008 [110]),

- over conflict resolution, and release planning

  > "...information about priorities is needed, not just so as to be able to ignore the least important requirements but also to help the project manager to resolve conflicts, plan for staged deliveries, and make necessary trade-offs."(Lehtola et al., 2004 [234]),

- over cost control

  > "Hence, the most cost effective way of developing software is to find the optimal set of requirements early, and then develop the software according to this set."(Berander and Andrews, 2005 [53])

- over preparing and easing decisions in subsequent development phases

  > "In current practice, software professionals need to make these trade-offs [*between requirements*] when selecting technologies being used, architectural patterns, and design solutions. These early decisions in the project have long-term and critical impacts on the product."(Elahi and Yu, 2011 [125])

  ,

- to balancing different qualities a system-to-be has to meet

  > "Although detailed information is typically scare during a project's early phases, developers frequently need to make key decisions about trade offs among quality requirements.McManus (2007 [266])" (Feather, Cornford, Hicks, Kiper, and Menzies, 2008 [139])

  .

In case that the topic of proper decision making and requirements selection is as important as it stated by the previously cited authors, one would expect evidence that the topic impacts software development projects in the "real world". Indeed such evidences seem to exist as researchers state based on industry projects, surveys and field studies:

> "Evidence suggests that the successful delivery of software engineering projects is directly related to the way in which stakeholders are involved in the decision making process within the various stages of the software development process."(Ognjanovic, Gasevic, Bagheri, and Asadi, 2011 [293])

> "Developing software systems that meet stakeholders' needs and expectations is the ultimate goal of any software provider seeking a competitive edge."(Karlsson and Ryan, 1997 [209])[6]

> "Requirements prioritization plays a crucial role in software development, and in particular it allows for planning software releases, combining strategies for budget management and scheduling, as well as market strategies."(Perini, Susi, and Avesani, 2013 [311]) [7]

[6] Also stated in a similar way by Svensson et al. (2011 [365]), Regnell et al. (2001 [319]), Berander and Andrews (2005 [53]), and Elahi and Yu (2011 [125]))

[7] Also stated in a similar way by Karlsson, Wohlin, and Regnell (1998 [210]), Li et al. (2010 [239]), and Tourwe, Codenie, Boucart, and Blagojevic (2009 [369]))

> "Software projects fail because of poorly selected and negotiated requirements which leads to inadequate products."(In, Olson, and Rodgers, 2002 [192]) [8]

In consequence, practitioners are demanding decision support in requirements engineering for a long time. Already in 1998, Karlsson et al. (1998 [210]) observed and documented these needs. But even up to now, decision supporting methods are not a regular part of best practice in requirements engineering as Svensson et al. (2011 [365]) observed in a survey conducted in eleven companies. This is seconded by an experience report by Tourwe et al. (2009 [369]) who state:

> "Given this relevance, it is striking to observe that the studied companies often implement fragmentary and inconsistent solutions for defining releases, leading to suboptimal and unsatisfactory results."(Tourwe et al., 2009 [369])

Hence, even though the importance of decision making in RE and requirements selection is known, there seems to be a lack of coherent and applicable solutions.

Within the field of decision making in RE and requirements selection there are three groups of topics which are seen by the researchers and practitioners in the field as separate topics. First of all, for making decisions about requirements these requirements need some kind of value for the stakeholders which is influenced by the preferences of the stakeholders regarding their goals, and to which extent these goals are satisfied by certain requirements (Firesmith, 2004 [146]). But a requirement might not only have a value for the end users but also for the software developer (Firesmith, 2004 [146]), because, for example, a software company tries to maximize the revenue of a system-to-be under the given time, effort, and budget constraints. Of course, a requirement can have different values regarding different aspects, for example, different qualities or constrains such as cost, time, effort. How to find those values and how to order requirements accordingly is treated in the field of *valuation and prioritization of requirements (short valuation)*.

But even if the values and according priorities are known, further decisions are necessary:

> "Once this is done [the prioritization] it may seem to be a fairly straightforward task to select requirements from the priority list until the total estimates equals the available resources. However, at the time of prioritization it is difficult to be fully aware of the context and circumstances present at release planing and hence further judgment is needed."(Carlshamre, 2002 [95])

Hence, there are further aspects beside the pure value which have to be considered for selecting requirements. One important aspect are the relations between requirements:

> "Furthermore, in a previous study (Carlshamre, Sandahl, Lindvall, Regnell, and Natt och Dag, 2001 [96]) it was found that close to 80% of the requirements had interdependencies pertinent to release planning."(Carlshamre, 2002 [95])

Such dependencies, which are in the simplest case requirements requiring or denying each other, make it difficult to select the optimal set of requirements right away. Hence, another topic of relevance is the *optimization of requirements* in which one tries to take all aspects including the values of requirements into account and to find optimal set(s) of requirements regarding defined dimensions, such as stakeholder satisfaction or return on investment (Perini et al., 2013 [311]; Li et al., 2010 [239]; Lehtola, Kauppinen, Vähäniitty, and Komssi, 2009 [235]; Lehtola and Kauppinen, 2006 [233]).

Additionally, there is also research on general *decision* theory in requirements engineering which focuses on how decisions are made, the human aspects in requirements engineering as

[8] Also stated in a similar way by Regnell et al. (2001 [319]), and Aurum and Wohlin (2003 [22]))

well as negotiation processes for finding compromise solutions (In et al., 2002 [192]; Regnell et al., 2001 [319]; Aurum and Wohlin, 2003 [22]).

In this thesis, we take a look at all of these topics, try to understand which activities and information are needed for selecting requirements, which the challenges are, how an ideal solution for selecting requirements would look like, and where the current state of the art is deficient regarding the ideal solution. In the next section, we introduce the according main research questions (MRQ) which guided us throughout our research.

1.3. Research Questions

Considering the importance of requirements engineering in general and the decisions taken for selecting the requirements to be addressed by a system-to-be in particular, the first and central main research question (MRQ) addressed by this work is:

> **MRQ1** How to select a set of requirements for a system-to-be in a way it satisfies the stakeholders' needs regarding the system-to-be to the maximum (optimal) possible extent?

But as we already have discussed, the decisions taken when selecting requirements and the correctness of such decisions are strongly dependent on the available information. Hence, to answer MRQ1 we have to go one step backwards in order to understand which information is necessary for selecting requirements:

> **MRQ2** Which information has to be known for an informed requirements selection?

But it is not only necessary to know which information should be collected and known in an ideal case. It is also of relevance if it is possible to collect this information within the RE activities and if there is guidance for collecting this information:

> **MRQ3** How to obtain the information necessary for requirements selection, and are there best practices available or are there any significant gaps?

While investigating MRQ3, we revealed some gaps regarding available methods to obtain the required information[9]. Two significant ones are investigated in depth in this work, namely:

> **MRQ4** How to elicit all relevant domain knowledge including the stakeholders and the context of the system-to-be in a structured way?

> **MRQ5** How to address compliance with laws and according qualities such as security and privacy when eliciting the requirements for a system-to-be?

1.4. A Process for Requirements Selection

Based on a problem & gap study on the matter of decision making and requirements selection, we show in this section which kind of information is important for requirements selection (Section 1.4.1), and which challenges one has to face when forming a solution for decision making and requirements selection (Section 1.4.2[10]). Finally, we propose a general process for collecting all this information and conducting the requirements selection (Section 1.4.3[11]).

[9] The importance of this information is discussed in the next section, while the found gaps are discussed in Chapter 6 (Page 81) and Chapter 14 (Page 239).

[10] Page 15

[11] Page 19

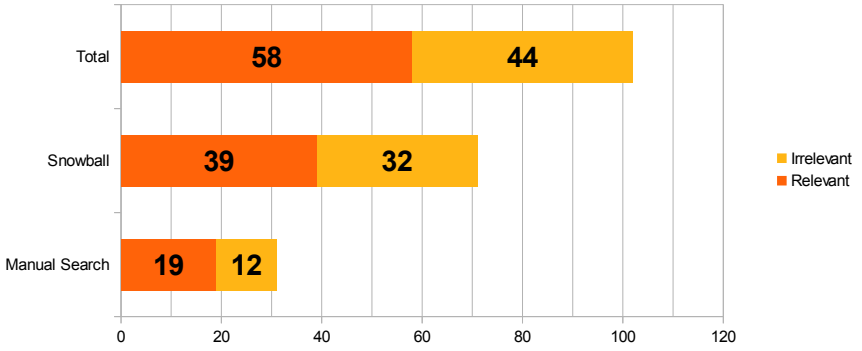


Figure 1.1.: *Found Literature*

1.4.1. Necessary Information

The starting point to shape a process which enables us to answer MRQ1 and MRQ3 is the information which has to be taken into account when selecting the requirements for a system-to-be and therefore has to be collected. Because the process has to give guidance on collecting the information and the decision support has to consider the collected information. Hence, we start with answering MRQ2.

For this purpose, we conducted a problem & gap study[12] about decision making and requirements selection in requirements engineering. Insights from the analysis of the found literature which are related to the information necessary for an informed requirements selection are presented in this section. The problem & gap study was conducted using a manual search in the publications of selected conferences and workshops spanning the years of 2009 until today[13]. The initially found literature was extended by snowballing[14] without any limit. For more details see Chapter 4[15].

Some numbers regarding the search are shown in Figure 1.1. After scanning the titles and abstracts of more than 8000 papers found while searching the selected venues, 31 papers were selected as potentially related to decision making and requirements selection in requirements engineering. Additionally, 71 were included after analyzing the bibliography of already included papers (snowballing). After reading the full papers, 19 papers of the manual search and 39 found while snowballing were selected as relevant papers. Hence, we found 58 relevant papers in total[16].

Within the literature on decision making in RE and requirements selection we could identify three main topics (see Figure 1.2). The first topic is about deciding on the value of a requirement (*valuation*). A value of a requirement in this context can be a single value regarding, for example, one quality such as security or business benefit, but also a set of values regarding different, for example, qualities and stakeholders. Many of the 28 papers (blue area in Figure 1.2) on this topic use prioritization techniques for this purpose. Almost 50% of the papers on decision making and requirements selection are on valuation of requirements. But still, this is only a preparatory step as even though the value(s) for each requirement is (are) known, the selection of requirements depends on more information than the pure value / priority such as, for example, the dependencies between requirements, as we will see later. The other half of the papers is on the actual decision about which requirements to address and which requirements to neglect

[12] see Section 3.2.6 (63) for a definition.
[13] The last iteration was done in March 2015
[14] The activity of analyzing the bibliography of an paper already identified as relevant to find further candidates is called snowballing.
[15] Page 67
[16] The list of relevant papers and the information collected for them is shown in the Appendix A.3.1

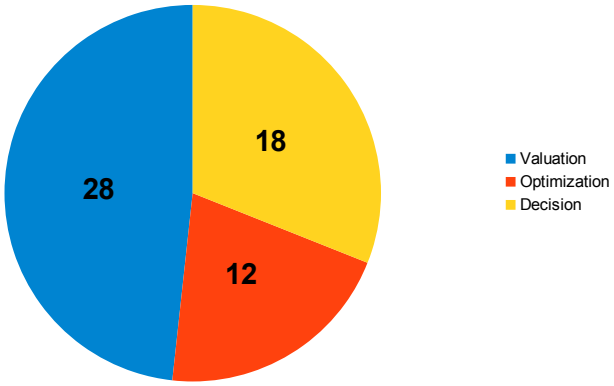


Figure 1.2.: *Decision Literature Topics*

when developing the system-to-be. 18 of these papers are on the decision processes, foundations, important factors, necessary negotiations and so forth (yellow area in Figure 1.2). They deal with the final decision, how to reach this decision, and the broader human context relevant for decision making (*decision*). An important specific means to reach a decision are *optimization* techniques, which directly calculate one or more optimal set(s) of requirements with regards to a decision model and the available information. 12 papers are on this topic (orange area in Figure 1.2). Hence, there is literature on making decisions in RE in general, how to decide on the value of requirements, and how to actually select an optimal set of requirements.

The topic of decision making in RE and requirements selection became a hot topic starting in the early 2000s (see Figure 1.3). The topic of valuation is the dominant topic in almost all years and well covered with literature, which is not surprising as it is the required prerequisite for selecting requirements. But also the topics of taking decisions in RE and using optimization techniques for this purpose are constantly represented in literature. Recently, there seems to be a small dip in for all three topics, but this might be due to the fact that the set of papers found during the manual search was quite small and the rest of the papers were found using backwards snowballing. Hence, the recent years might be a little bit underrepresented in the selected papers.

Up to this point, we can conclude that for the field of decision making in RE and requirements selection, the topics and activities of "general decision theory and decision processes", "valuating requirements", and "using optimization techniques for selecting requirements" are of central relevance. So is the information which is relevant for and used within the solutions proposed by the papers on these topics. But still, the question remains which information is considered to be important and necessary for decision processes, valuation of requirements and optimization of requirements.

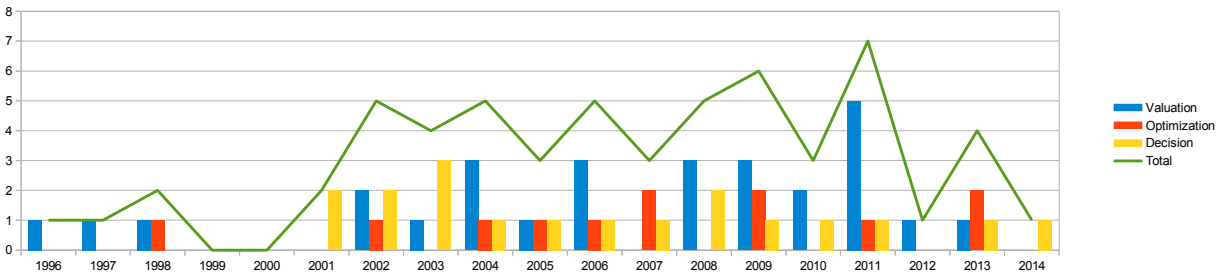


Figure 1.3.: *Distribution of Papers Over Time*

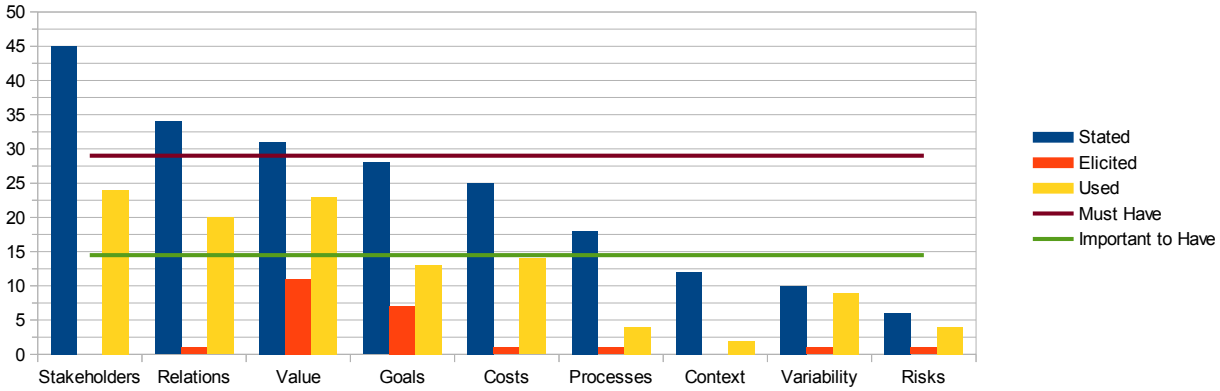


Figure 1.4.: *Information Necessary for Requirements Selection*

To answer this question, we analyzed the found papers for statements which highlight the importance of a certain type of information, which information is actually used within proposed solutions, and for which information the elicitation is described or at least an elicitation method is referenced. Figure 1.4 shows the results of this analysis. We listed all kinds of information which was stated (blue bars in Figure 1.4) by at least 5 different papers. The yellow bars indicate the actual number of solutions using a specific kind of information, while the orange bars indicate the number of papers which actually explain how to elicit this information.

We consider those kinds of information as *must have* if they are stated in the majority of papers (more than half of the papers (29)), because for this information we have a strong indication of its importance. Hence, "must have information" is information about the relevant *stakeholders*, which have to be taken into account when selecting requirements, the relations between requirements such as, for example, dependencies and conflicts, and the value of requirements for, for example, the different stakeholders.

Then there are kinds of information which should be collected and which are *important to have* for an informed decision. Such kinds of information are stated by more than 1/4 (14.5) of the papers, which we consider as a medium indication of importance. Such kinds of information are goals, costs, and processes to be supported by a system-to-be. Goals relate stakeholders and the values of requirements in more detail, because the value of a requirement for a specific stakeholder is related to the goals this stakeholder wants to have fulfilled by the system-to-be. In consequence, making the goals explicitly visible and taking them into account increases the reliability of the valuation as well as the optimization, because it clarifies what is valuated for a requirement, and which goals are the target of an optimization. The information about cost covers all constraints we might face when developing the system-to-be, such as, for example, time, money, and effort constraints. But costs not only mean development costs but also costs for the end user, which can be money to be spent for the system-to-be, effort for running it, and so forth. Costs are of importance, because when developing a system-to-be we do not only have to take the positive value(s) (benefits) of a requirement into account but also its impact on resources, which are limited for a development of a system-to-be. In this sense, costs are negative values. Processes are of importance, because they imply further relations between requirements. For example, requirements necessary to implement a specific process are dependent on each other as missing one requirement would break the process, rendering the rest of the requirements unnecessary. Additionally, it is likely that functional requirements related to the same process are complemented by the same qualities. Processes also contain information about the requirements which have to be fulfilled in parallel by the system-to-be, which is of importance for analyzing feature interactions. Hence, processes are useful for finding

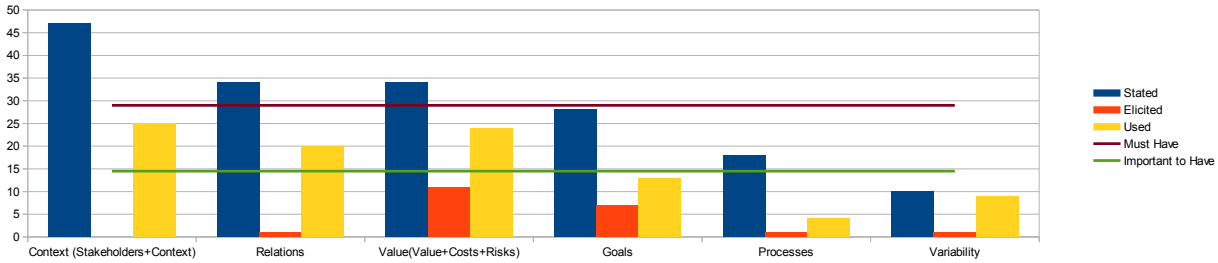


Figure 1.5.: *Information Necessary for Requirements Selection (Aggregated)*

and detailing the relations between requirements.

A third group of information is only stated by less than 1/4 of the papers. We assume that these kinds of information are *nice to have* or only necessary under certain circumstances. The first "nice to have" information is the knowledge about the context of the system-to-be beside the stakeholders. For decision making the stakeholders are of course of central relevance, but also other entities in the environment of the system-to-be such as, for example, existing other systems might play a role in the decision process. We observed for the context information that many authors implicitly assume that such domain knowledge is already known and codified into the goals of stakeholders or the requirements themselves accordingly. Hence, here the importance might be higher than implied by the explicit statements raised within the papers. Another "nice to have" information is information about variability in requirements. It is acknowledged by many authors that adding variability to requirements increases the magnitude to which possibly conflicting needs of different stakeholders can be addressed. But still, it is seen as a special area within requirements engineering to handle and use variability, for example, by introducing software product lines. The third kind which is stated by some papers is the information about risks. But risk management is mostly seen as source for one more cost value.

We summarized and aggregated the results of the problem & gap study regarding the information necessary for a informed requirements selection as follows (see Figure 1.5)[17]:

Context [18] From our point of view stakeholders are part of the context for the system-to-be. Hence, we need information about this context. But the context is not limited to the stakeholders, as we will discuss in Chapter 6[19], but covers all important domain knowledge including also existing systems, interfaces and so forth. As decisions are always regarded as context sensitive and only correct with respect to a given context, which is also indicated by the results of the problem & gap study, the information of the context is crucial for an informed requirements selection. Additionally, as already discussed in Section 1.1[20], this domain knowledge is of crucial importance for a successful RE process itself. Hence, collecting context information is essential not only for the requirements selection.

Relations [21] The relations between requirements are also of central importance when selecting requirements, because they constrain the possible sets of requirements we can choose

[17] Some references for each information type are given. These references discuss the according type in detail. Note that such references given are not complete. A complete overview of papers stating an information kind can be found in Appendix A.3.1

[18] (Regnell et al., 2001 [319]; Carlshamre, 2002 [95]; Ruhe, Eberlein, and Pfahl, 2002 [324]; Daneva and Herrmann, 2008 [110]; Berander and Andrews, 2005 [53]; Ruhe, Eberlein, and Pfahl, 2003 [325]; Lehtola and Kauppinen, 2006 [233]; Horkoff, Salay, Chechik, and Di Sandro, 2014 [187]; Saliu and Ruhe, 2005 [335]; Ruhe et al., 2003 [325])

[19] Page 81

[20] Page 3

[21] Firesmith (2004 [146]), Finkelstein, Harman, Mansouri, Ren, and Zhang (2009 [144]), Li et al. (2010 [239]), Ognjanovic et al. (2011 [293]), Tourwe et al. (2009 [369]), Lehtola et al. (2009 [235]), Carlshamre (2002 [95]),

from. Conflicts deny the parallel selection of certain requirements, while dependencies require the parallel selection of other requirements. Conflicts and dependencies are the strongest kinds of relations regarding their impact on the selection process, but there are also different shades of conflict and dependency possible for requirements relations. For example, when talking about quality requirements, one quality requirement might only hinder the fulfillment of another quality requirement, but not completely deny it. In general, the relations between requirements are seen as one of the factors which make the requirements selection complicated in the end, because they dictate which requirements solutions are implementable as well as they might also influence the final outcome regarding the sufficiency for the different stakeholders.

Values Of course, when selecting requirements they are selected according to the value they spent for the different stakeholders. To satisfy a stakeholder the system-to-be has to provide some benefit to him / her[22]. Such benefits can have multiple dimensions, for example, regarding different desired qualities, business values, and so forth. The benefits are the values we want to maximize for all stakeholders to satisfy the stakeholders to the maximum possible extent. But from our point of view benefits are only one group of values a requirement has. The other group are the liabilities a requirement bears, such as, for example, implementation costs[23], implementation time[24], effort[25], risks they introduce[26], and so forth. Those liabilities constrain the possible sets of requirements we can choose from, because we often face, for example, limited budgets, limited time spans for implementation, and limits regarding the acceptable risks. Hence, the liabilities are another factor which complicates the selection of requirements.

Goals [27] Goals are necessary to explicitly state what the different stakeholders expect from a system-to-be, and which different benefits a requirement entails are of relevance for a decision as well as they are the central means for the optimization and selection. But still, they are considered to be only "nice to have" because many papers only treat the goals implicitly since they are hidden in, for example, a general aggregated value for a requirement or are part of the negotiations but never stated and captured explicitly. Hence, one can benefit from having the goals explicitly stated as they make the decisions more transparent, but they are not a "must have".

Regnell et al. (2001 [319]), Daneva and Herrmann (2008 [110]), Berander and Andrews (2005 [53]), van den Akker, Brinkkemper, Diepen, and Versendaal (2008 [376]), and Saliu and Ruhe (2005 [335])

[22] Firesmith (2004 [146]), Li et al. (2010 [239]), Ognjanovic et al. (2011 [293]), Akker, Brinkkemper, Diepen, and Versendaal (2005 [3]), Daneva and Herrmann (2008 [110]), Herrmann and Daneva (2008 [183]), Lehtola et al. (2004 [234]), Berander and Andrews (2005 [53]), van den Akker et al. (2008 [376]), Saliu and Ruhe (2005 [335]), and Perini, Ricca, and Susi (2009 [310])

[23] Firesmith (2004 [146]), Perini et al. (2013 [311]), Karlsson et al. (1998 [210]), Li et al. (2010 [239]), Akker et al. (2005 [3]), Liaskos, Lapouchnian, Yu, Yu, and Mylopoulos (2006 [240]), Herrmann and Daneva (2008 [183]), Berander and Andrews (2005 [53]), and Elahi and Yu (2011 [125])

[24] Firesmith (2004 [146]), Perini et al. (2013 [311]), Karlsson et al. (1998 [210]), Li et al. (2010 [239]), Saliu and Ruhe (2007 [334]), Tourwe et al. (2009 [369]), Akker et al. (2005 [3]), Lehtola et al. (2004 [234]), van den Akker et al. (2008 [376]), and Saliu and Ruhe (2005 [335])

[25] Firesmith (2004 [146]), Perini et al. (2013 [311]), Saliu and Ruhe (2007 [334]), Ruhe et al. (2002 [324]), Regnell et al. (2001 [319]), Herrmann and Daneva (2008 [183]), Lehtola et al. (2004 [234]), Berander and Andrews (2005 [53]), Saliu and Ruhe (2005 [335]), and Ruhe et al. (2003 [325])

[26] Firesmith (2004 [146]), Li et al. (2010 [239]), Elahi and Yu (2011 [125]), van den Akker et al. (2008 [376]), and Saliu and Ruhe (2005 [335])

[27] Firesmith (2004 [146]), Li et al. (2010 [239]), Saliu and Ruhe (2007 [334]), Ognjanovic et al. (2011 [293]), Tourwe et al. (2009 [369]), Akker et al. (2005 [3]), Liaskos et al. (2006 [240]), Ruhe et al. (2002 [324]), Liaskos, McIlraith, Sohrabi, and Mylopoulos (2011 [241]), Lehtola et al. (2004 [234]), Berander and Andrews (2005 [53]), van den Akker et al. (2008 [376]), Saliu and Ruhe (2005 [335]), Perini et al. (2009 [310]), and Ruhe et al. (2003 [325])

Processes [28] Whenever a system-to-be is process driven, representing the processes explicitly and relating the requirements to these processes is considered to be necessary by many researchers. The relations between processes and requirements allows one to assure that the functionality of a system-to-be covers all required processes, but also helps to derive relations between requirements. Hence, when a system-to-be has to follow certain processes, those processes impose constraints on the requirements selection. But as not all systems are process driven or the processes are not the driving factor for choosing requirements and solutions, the information about processes is important to have as it is beneficial in all cases, but it is not a must have.

Variability [29] Information about the variability of requirements can be helpful when optimizing the requirements selection, because different variants for one requirement increase the flexibility regarding the constraints to be met. For example, while two requirements which constitute a high value might exclude each other, a variant for one of the requirements might resolve this conflict. Hence, both requirements (one as a modified variant) can be added to the selected requirements. But still, in many cases such variants cannot be derived for the requirements of a system-to-be or deriving them is too costly. In consequence, the information about requirements variants is nice to have.

In this section we have answered MRQ2:

> **MRQ2** Which information has to be known for an informed requirements selection? *The information about the context of the system-to-be, the relations between the requirements, and the values assigned to each requirement have to been known for a requirements selection. Information about goals, processes, and variants further improve the selection of requirements.*

1.4.2. Further Insights From a Problem & Gap Study in the Field of Decision Making and Requirement Selection in RE

Beside the information required for an informed decision, we gained some further insights while conducting our problem & gap study. These insights and the derived challenges are discussed in the following. We used the list of challenges stated by Firesmith (2004 [146]), which is widely cited in literature, a field study of Lehtola and Kauppinen (2006 [233]), and an interview study of Kukreja, Payyavula, Boehm, and Padmanabhuni (2012 [227]) as starting point. The resulting list of challenges was used to analyze the rest of the found papers and finally augmented and extended using the gained insights.

The first group of problems we identified is related to the valuation of requirements, and these problems are crucial as the valuation of requirements is considered to be a very expensive task, in terms of effort spent, within the requirements selection process (Perini et al., 2013 [311]). The challenges related to this topic are:

Relevant aspects for valuating requirements unknown One problem is that stakeholders do not exactly know which preferences they have and which goals are relevant to them:

> "As we will see in the rest of this paper, in real world software decision making scenarios, the stakeholders do not have a clearcut and well understood preference over all the available options." (Ognjanovic et al., 2011 [293])

[28] Ma, Liu, Xie, Zhang, and Yin (2009 [247]), Liaskos et al. (2011 [241]), Laurent, Cleland-Huang, and Duan (2007 [232]), and Liaskos et al. (2006 [240])

[29] Feather et al. (2008 [139]), Liaskos et al. (2011 [241]), Elahi and Yu (2011 [125]), and Heaven and Letier (2011 [177])

> "Companies often do not know exactly what the relevant criteria are for expressing value in their context."(Tourwe et al., 2009 [369])[30]

Hence, it is necessary to firstly identify what has to be valuated for a requirement, before the valuation takes place (Lehtola et al., 2004 [234]; Berander and Andrews, 2005 [53]; Elahi and Yu, 2011 [125]; Lehtola and Kauppinen, 2006 [233]; Carlshamre, 2002 [95]; Saliu and Ruhe, 2005 [335]). Without an explicit definition of what has to be valuated, stakeholders are not able to give a proper value for a requirement at all (Lehtola et al., 2009 [235]).

Inconsistent judgments Even when it is known what has to be valuated the problem remains that subjective judgments are not consistent, for example, transitive in all cases:

> "The consistency check is very important since human judgment is far from perfect. The absence of a consistency index may make the process unreliable since nobody can assess the potential judgmental errors being made."(Karlsson et al., 1998 [210])[31]

Hence, a valuation method must be able to handle such inconsistencies.

Absence of quantitative data Even for aspects which can be quantified easily in general, such as costs, in the early phases of software engineering, such as RE, this information is often unknown (Lehtola et al., 2009 [235]; Lehtola et al., 2004 [234]; Elahi and Yu, 2011 [125]; Lehtola and Kauppinen, 2006 [233]; Saliu and Ruhe, 2007 [334]). Hence, there needs to be a way to valuate requirements without requiring one to provide concrete quantitative data.

Uncertainty of value Many judgments and values given by stakeholders are far apart from being stated with 100% certainty (Tourwe et al., 2009 [369]; Horkoff et al., 2014 [187]; Letier, Stefan, and Barr, 2014 [237]). This has to be taken into account when proposing a requirements selection process.

Volatile values Preference might change over time. In consequence, also the values obtained might change (Firesmith, 2004 [146]; Akker et al., 2005 [3]; Kukreja et al., 2012 [227]; Ruhe et al., 2003 [325]; van den Akker et al., 2008 [376]). Such a change should be anticipated and reflected within a method. Reruns because of changed values should be possible while spending a significantly lower effort compared to the initial run.

Multidimensional values The value of a requirement is in most cases multidimensional (Lehtola et al., 2004 [234]; Perini et al., 2013 [311]; Li et al., 2010 [239]). As we have already seen in Section 1.4.1[32] there are several, in some cases competing, goals as well as as constraints of different kinds. Hence, a decision making process has to reflect all these values and enable one to balance them properly.

Unknown stakeholders and stakeholder preferences But even when the valuation itself reflects all the problems above, the problem remains who the stakeholders are, who should be asked to state their preferences (Svensson et al., 2011 [365]). Another question is how to get their preferences, because in some cases they are not directly available or only in a restricted way (Lehtola et al., 2004 [234]). Hence, a valuation method has to be flexible and easy to adapt to different value elicitation scenarios, such as interviews, surveys and so forth.

A second group of problems is related to the requirements, their relations, and how they are expressed:

[30] A similar observation is made by Lehtola et al. (2009 [235])
[31] Seconded by Firesmith (2004 [146])
[32] Page 9

Mandatory requirements When making decisions and selecting requirements, it has to be clear that some requirements cannot be subject of a decision as they are mandatory (Firesmith, 2004 [146]; Wiegers, 2000 [387]; Liaskos et al., 2011 [241]; Kukreja et al., 2012 [227]; van den Akker et al., 2008 [376]). This has to be reflected by a requirements selection method.

Kinds of requirements Such a method has also to take into account that there are different kinds of requirements, such as functional and quality requirements of different kinds (security, performance, and so forth) (Firesmith, 2004 [146]; Feather et al., 2008 [139]; Karlsson et al., 1998 [210]; Ognjanovic et al., 2011 [293]; Trummer, Faltings, and Binder, 2014 [371]; Svensson et al., 2011 [365]; Regnell et al., 2001 [319]; Daneva and Herrmann, 2008 [110]; Herrmann and Daneva, 2008 [183]; Berander and Andrews, 2005 [53]; Elahi and Yu, 2011 [125]; Ruhe et al., 2003 [325]; van den Akker et al., 2008 [376]). This has to be reflected. The current state of the art seems to neglect this fact and treats all kinds of requirements in the same way. Especially qualities do not seem not be treated with care:

> "Prioritization of quality requirements warrants further attention. Not a single study looked into techniques of prioritization of quality requirements".(Svensson, Host, and Regnell, 2010 [364])[33]

Reasons for these observations are, according to a study of Svensson et al. (2011 [365]), that the elicitation of quality requirements is hard, the specification of quality requirements is complicated, and the knowledge about desired qualities is missing. In the end, Svensson et al. (2011 [365]) state that

> "If the quality requirements are well described, they are as easy as functional requirements to prioritize."(Svensson et al., 2011 [365])

Hence, a process for requirements selection has to take care that it is known which qualities are desired, that those qualities are reflected in according requirements, and that these requirements are described well.

Legal requirements One specific kind of requirements, which are also often mandatory requirements, are legal compliance requirements. Such requirements implied by laws which are relevant for the system-to-be have a great impact on decisions taken and requirements selected, but are hard to elicit and intertwine with other requirements of the system-to-be (Firesmith, 2004 [146]; Wiegers, 2000 [387]; Lehtola et al., 2004 [234]; Kukreja et al., 2012 [227]; Massey, Otto, and Antón, 2009 [255]). Hence, a process for requirements selection should explicitly take care of the legal compliance topic.

Number of requirements As the systems nowadays get more complex as well as the environment they are used in, the number of requirements tends to get high. Hence, a solution for requirements elicitation and selection not only has to work for small scale systems but also for large systems (Firesmith, 2004 [146]; Li et al., 2010 [239]; Kukreja et al., 2012 [227]; van den Akker et al., 2008 [376]; Greer and Ruhe, 2004 [162]; Lehtola and Kauppinen, 2006 [233]).

Relations between requirements To discover relations between requirements is not an easy task. For example, to detect unwanted interactions between requirements, which lead to a requirements conflict, is a research field on its own for a long time but still interaction detection is a challenge. Especially for quality requirements only few solutions exist. But even

[33]Seconded by Svensson et al. (2011 [365]), and Firesmith (2004 [146])

the fact that a requirement is dependent on another requirement might not be easily noticed as such dependencies might be hidden in, for example, complex processes. Hence, discovering the relations between requirements and address them accordingly is seen as important challenge (Tourwe et al., 2009 [369]; Finkelstein et al., 2009 [144]; Li et al., 2010 [239]; Saliu and Ruhe, 2007 [334]; Ognjanovic et al., 2011 [293]; Lehtola et al., 2009 [235]; Firesmith, 2004 [146]; Carlshamre, 2002 [95]; Daneva and Herrmann, 2008 [110]; Berander and Andrews, 2005 [53]; Kukreja et al., 2012 [227]; Saliu and Ruhe, 2005 [335]). Still, there seems to be a gap regarding the importance of relations and actual solutions taking them into account:

> "We could not find any activity or method which is specifically designed for coping with dependencies."(Daneva and Herrmann, 2008 [110]; Herrmann and Daneva, 2008 [183])

Relation between goals and requirements Goals are conceptually on a higher level of abstraction than requirements. Additionally, goals are stakeholder-centric, while requirements are system-centric. Therefore, refining requirements taking into account more detail of the system-to-be and analyzing the system-to-be described by the requirements is reported to be difficult for goal-oriented methods (Alrajeh, Kramer, Russo, and Uchitel, 2009 [14]). But the relations between goals and requirements are of central importance, as the goals determine why a requirement should be selected (Firesmith, 2004 [146]; Kukreja et al., 2012 [227]). Hence, it is important to relate goals and requirements.

Alternatives Especially when treating conflicting requirements it is often an option to introduce alternative requirements which mitigate or at least lessen the conflict. Such alternatives might not bring in the full benefit like the original requirements, but enable one to address both original requirements to some extent. Hence, alternative generation and consideration is an important tool in requirements decision making (Feather et al., 2008 [139]; Liaskos et al., 2011 [241]; Elahi and Yu, 2011 [125]; Heaven and Letier, 2011 [177]), but is also a neglected topic in requirements selection:

> "However, while those approaches address priority elicitation, the problem of modeling and reasoning about priorities and alternative solutions has not received much attention."(Liaskos et al., 2011 [241])

Volatile requirements Requirements might change over time (Firesmith, 2004 [146]; Akker et al., 2005 [3]; Kukreja et al., 2012 [227]; Ruhe et al., 2003 [325]; van den Akker et al., 2008 [376]). Such a change should be anticipated and reflected within a method. Reruns because of changed requirements should be possible while spending a significantly lower effort compared to the initial run.

Comparable requirements When asking to decide which requirements to select or to valuate requirements, it is a big problem when requirements are not expressed on the same level of abstraction, not expressed using a coherent wording and notation, and the complexity of requirements differs to a large extent (Lehtola and Kauppinen, 2006 [233]). Hence, all requirements have to be aligned within a requirements decision and selection process regarding the used wording, the used notation, the level of abstraction / detail, and their complexity.

A third group of challenges considers the decision process itself:

Decision making as political process Several researchers consider the decision process to be also a political process (Aurum and Wohlin, 2003 [22]; In et al., 2002 [192]; Regnell et al., 2001 [319]; Laurent et al., 2007 [232]). The reason is that different stakeholders might

have conflicting views on the system to be and in consequence also competing requirements (Tourwe et al., 2009 [369]; Saliu and Ruhe, 2007 [334]; Ognjanovic et al., 2011 [293]; Firesmith, 2004 [146]; Berander and Andrews, 2005 [53]; Horkoff et al., 2014 [187]; Aurum and Wohlin, 2003 [22]; Laurent et al., 2007 [232]; Ruhe et al., 2003 [325]). Hence, the different views and requirements need to be balanced and negotiated until a compromise solution is found (In et al., 2002 [192]; Regnell et al., 2001 [319]; Aurum and Wohlin, 2003 [22]).

Decision making as collaborative process The process of collecting all the necessary information, aggregating it and finally selecting requirements requires different stakeholders as well as the requirements engineers to collaborate (Firesmith, 2004 [146]; Wiegers, 2000 [387]; Regnell et al., 2001 [319]; Bagheri, Asadi, Gasevic, and Soltani, 2010 [29]; Lehtola and Kauppinen, 2006 [233]; Saliu and Ruhe, 2005 [335]; Aurum and Wohlin, 2003 [22]). As a result, a process should clearly state what has to be done by whom in which way. It should also guide the collaboration itself.

Flexibility of the decision process A decision making and requirements selection process should be modular (Maiden and Rugg, 1996 [250]). This means that complete activities can be skipped without breaking the complete process as well as being able to replace certain methods used for an activity by another method (Kukreja et al., 2012 [227]). The need for such flexibility lies in the fact that the available methods change and improve over time, and one should be able to profit from the progress made. Another reason is that some steps might only be of importance under certain circumstances or if enough resources are available to conduct them. For example, if the business processes are not relevant for the system-to-be we do not want be forced to still describe processes.

Documentation and aggregation of information To prepare the requirements selection a large amount of information is collected. To document all this information and to aggregate information from different sources can be challenging (Lehtola et al., 2004 [234]; van den Akker et al., 2008 [376]). Hence, a requirements selection process should clearly state the required information, how to document it, and, in case there is more than one source for an information, how to aggregate information.

Decision traceability In the end not only the decisions themselves are of importance, but also the ability to make the reasons for a decision transparent (Liaskos et al., 2011 [241]; Regnell et al., 2001 [319]; Kukreja et al., 2012 [227]; Lehtola and Kauppinen, 2006 [233]). Hence, there needs to be traceability between all the artifacts important for a decision and the decision itself.

Ease of use The complete decision making and requirements selection process is complex, involving many different participants. These participants have very different backgrounds and also time to spend on the process. Hence, it is necessary to tailor the process in a way that it is as easy to learn and use for each participant as possible (Kukreja et al., 2012 [227]; Bagheri et al., 2010 [29]).

Preexisting Domain Knowledge Context is important for decision making, but a process should not require the participants to have a broad domain knowledge before entering the process (Kukreja et al., 2012 [227]; Lehtola and Kauppinen, 2006 [233]; Ruhe et al., 2003 [325]). Hence, within the process one has to assure that the needed domain knowledge is elicited and communicated to those participants who need it.

Tool support The activities within decision making include information elicitation, information documentation, coherence checking, information aggregation, information transformation, formal reasoning, calculations and so forth. To conduct all these things by hand is near

to impossible. Hence, there is strong need for tool support (Svensson et al., 2010 [364]; Kukreja et al., 2012 [227]; Saliu and Ruhe, 2005 [335]).

1.4.3. Resulting Process

Based on the information required for an informed decision (Section 1.4.1[34]) and the further insights gained from a problem & gap study in decision making and requirements selection (Section 1.4.2), we propose a general process for collecting the important information for decision making and the actual requirements selection in the following. The process is shown in Figure 1.6. The process itself is split up into the three phases *understanding the purpose*, *understanding the problem*, and *reconciliation*[35].

This grouping of activities is in line with general decision theory. Aurum and Wohlin (2003 [22]) discuss several decision frameworks and their relation to RE processes. They relate the activities of such frameworks and define in the end also three groups of activities, which are *problem identification*, *problem analysis*, and *selection*. The problem identification according to Aurum and Wohlin (2003 [22]) is to get the context and the reasons why there is a problem somebody

[34] Page 9

[35] Note that Figure 1.6 also shows the intersection to the design activities (*intertwining requirements & architecture*), but the design of the system-to-be is not a topic in this thesis.

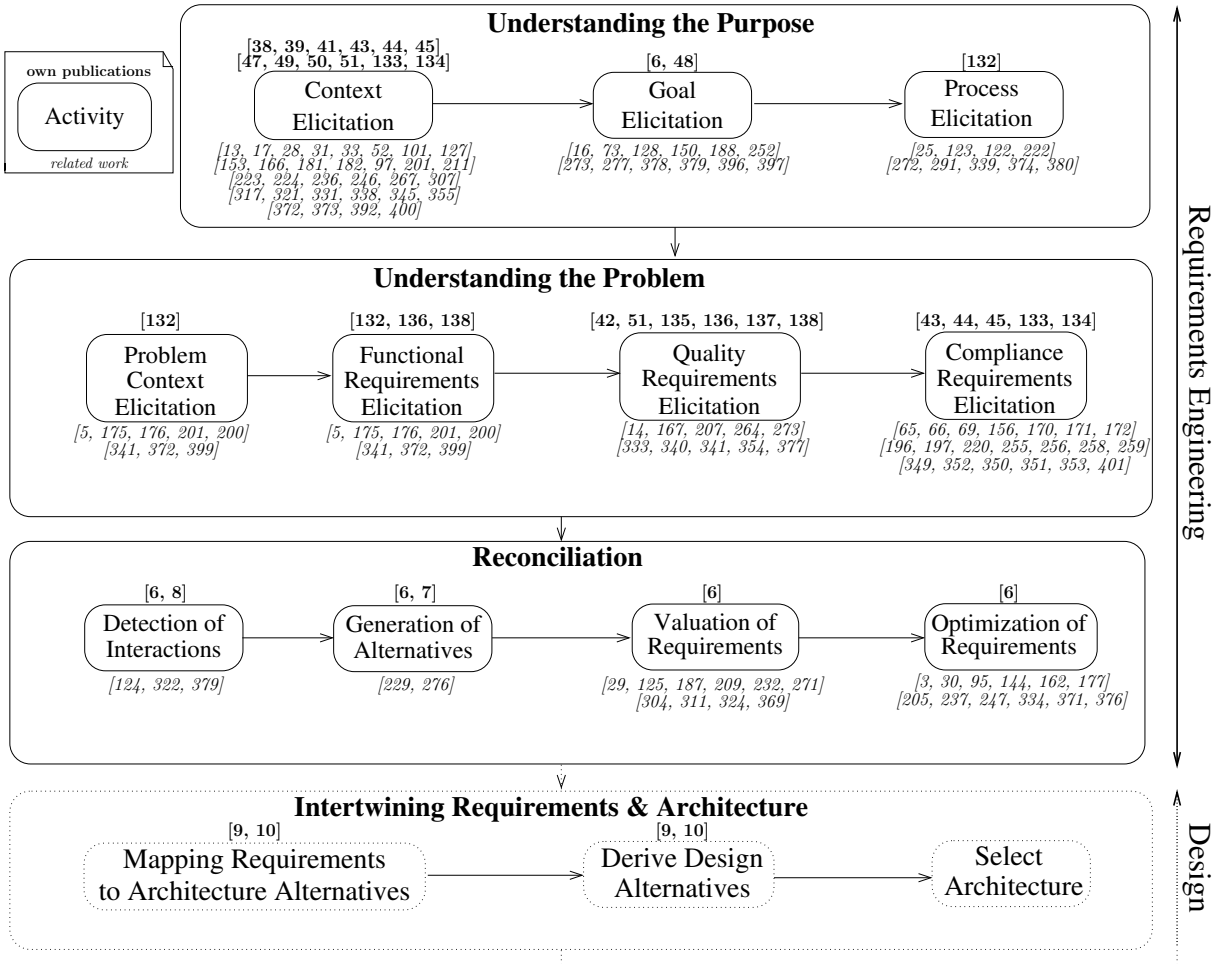


Figure 1.6.: *Overall Process*

wants to be resolved, the problem analysis is then to understand the details of the problem, and the selection is about the options to resolve the problem and how to select them. In terms of our process, problem identification is about understanding the purpose of the system-to-be, the problem analysis corresponds to the understanding of the problem, and selection is equivalent to our reconciliation phase.

Our proposed phases are also in line with the general idea of requirements engineering as defined by Zave (1997 [398]), and Nuseibeh and Easterbrook (2000 [290]), which is about the why, what, and which of a system-to-be (see Section 1.1[36]). Understanding the purpose is about the why of the system-to-be, understanding the problem is about the what of the system-to-be, and reconciliation about the which.

The activities (each activity shown in Figure 1.6 is annotated with of own publications on the matter (*bold* font) and related work (*italic* font)) of the process are explained briefly in the following (The description is based on Alebrahim, Choppy, Faßbender, and Heisel (2014 [6])[37]):

Understanding the Purpose The aim of this phase is to understand the purpose of the system-to-be. Thus, functionality is not considered for the moment, but the direct and indirect environment of the system-to-be. In this phase we want to collect all stakeholders, other already established systems, assets and other entities, which are directly or indirectly related to the system-to-be.

Context Elicitation The first step is to collect all relevant entities of the environment. Therefore, we use domain specific patterns, which help us to point out where to look for relevant information. This information is documented using graphical patterns, which are accompanied with textual templates. The documented information not only describes the relevant entities, but also captures their relation to each other and the system-to-be. Entities can be stakeholders, assets, systems, and so on.

Goal Elicitation When dealing with decision making, we always talk about decisions which are in line with certain goals. Hence, we need to know what the goals are to be considered for decision making. Therefore, we have to analyze for each stakeholder, known from context elicitation, which goals he/she has in relation to the system-to-be. Conducting goal elicitation before process elicitation is reasonable as the stakeholders barely have a complete overview of the process but exactly know what they want to achieve and have to do (Gröner, Asadi, Mohabbati, Gasevic, Parreiras, and Boskovic, 2012 [163]; Frankova et al., 2011 [151]; Singh and Woo, 2009 [355]). Hence, a goal model is a good starting point for discovering processes.

Process Elicitation Nowadays, it is crucial when developing software to consider the business processes which are supported by the system-to-be. Modern architectures try to follow the rule "Build software which follows the organization" in contrast to "The organization has to be adapted to the software". From the previous steps we already know all stakeholders and their goals. Now it is feasible to identify the related business processes and model them in detail.

Understanding the Problem In this phase, we aim at understanding the system-to-be, the problem it shall solve, and therefore understanding the environment it should influence according to the requirements.

Problem Context Elicitation In this step we elicit all domains related to the problem to be solved, their relation to each other and the machine. Here we get a full description of

[36] Page 3

[37] The RE part is a contribution of the author.

the environment and the machine within this environment. Thus, we have a complete problem description.

Functional Requirements Elicitation In the next step we decompose the overall problem into sub-problems, which describe a certain functionality described by a requirement. The functionality of the machine is the core and all quality requirements are related in some way to this core. Hence, having the basic functional requirements first and then going for other requirements seems to be reasonable.

Quality Requirements Elicitation In the third step of phase two, we have to elicit the quality requirements and relate them to functional requirements of the system-to-be.

Compliance Requirements Elicitation When we know which requirements should be fulfilled to get the ideal machine, we have to check whether this machine is compliant to the regulations it has to obey or not. Regulations are laws, standards, internal policies and so forth. The most important are the requirements which are derived from laws. Software engineers have to ensure that the system-to-be complies to all relevant laws. If they disobey some laws they must know which laws they did not consider and provide a reasoning. Legal requirements have a huge potential for interactions with other requirements. Hence, they have a signification impact on the selection of requirements.

Reconciliation In the reconciliation phase we have to derive a clean set of requirements, which are free of unresolved or unwanted conflicts but nevertheless solves the problems and therefore satifies the initial goals of the stakeholders.

Interaction Detection The initial step for reconciliation is to discover the interaction between the requirements. Interactions can be positive or negative. They range from requirements requiring other requirements to requirements denying other requirements. To detect such interactions is quite complex and different kinds of requirements have to be treated in different ways.

Generation of Alternatives When discovering interactions we keep the interactions unresolved for the moment. Resolving the conflicts to get the optimal set of requirements will be addressed in the optimization step. To ease the decision making and to get a resulting set of requirements, which is as near to the optimal solution for every stakeholder as possible, we need to generate alternatives for problematic requirements. Hence, an original requirement might be excluded from the final set, but a weaker variant of this requirement might be included in the optimal set of requirements.

Requirements Valuation For decision making, the requirements need at least one value to which the optimization can refer. Also the relations between requirements need to be valuated. The valuation measure or method has to be selected in this step and the values for the requirements have to be elicited.

Requirements Optimization In this step all the requirements, their relations and the corresponding values are the input to the optimization model, which has to be developed. The optimization model should then compute the optimal set of requirements regarding the optimization goals and the valuation of the requirements.

Note that the whole process is meant to be modular and agnostic to the methods actually used for an activity. Hence, not every step has to be conducted in the way we propose it or conducted at all. We will discuss in Chapter 20[38] how our modular optimization model is able to cope with missing information.

[38] Page 341

For our process we have chosen methods which are model-based and require one to have a comprehensive documentation. This might been seen skeptically especially under the light of agile development. However, there are striking arguments for this decision. First of all, requirements engineering is often not the activity which is part of the agile development itself. For example, in scrum the backlog (list of requirements) and the according priorities have to be known before the development starts. Hence, here we have no conflict with the agile manifest itself. Secondly, it has been shown that on the long run having means to store and share knowledge is necessary even in agile development (Sadraei et al., 2007 [332]; Maalej and Ghaisas, 2014 [248]). While developing, domain and application knowledge has to be spread. Documentation is inevitable for this purpose (Curtis, Krasner, and Iscoe, 1988 [108]; Maalej and Ghaisas, 2014 [248]). For example, a study of Liu, Li, and Peng (2010 [244]) has shown that imperfect documentation of requirements and domain knowledge is one of the reasons for failing requirements engineering efforts in Chinese software industry. After delivery of a product, having documentation is inevitable for good maintenance (Sadraei et al., 2007 [332]; Maalej and Ghaisas, 2014 [248]). A model-based process supports all of these activities.

In practice, natural language is another option used by a majority of companies, but using natural language has turned out to be imperfect (Pinto-Albuquerque and Rashid, 2014 [313]). In turn, models are assumed to be a good way to structure and abstract requirements to handle complexity, reduce the natural language defects, and enable tool support (Cheng and Atlee, 2007 [100]). Actually, a study within a global software vendor showed that models really help to handle complexity and allow different necessary analysis (Maalej and Ghaisas, 2014 [248]), which is seconded by other studies (Hutchinson, Rouncefield, and Whittle, 2011 [190]; Neill and Laplante, 2003 [281]).

Another strong point for our proposed methods for the activities of our process is that within requirements engineering there is a need to transform initial informal information into (semi-) formal specifications (Fuentes-Fernandez et al., 2010 [153]). The stakeholders of a system-to-be need high level representations which relate to their "world" in which the system-to-be will be placed. Hence, they are not in need of the full details needed for the actual development, but concepts they can easily understand and which allow a reasoning about what they want from the system-to-be. In contrast, developers require a detailed problem understanding which includes parts of the context relevant to the system-to-be.(Coughlan, Lycett, and Macredie, 2003 [106]; Coughlan and Macredie, 2002 [105]; Maalej and Ghaisas, 2014 [248]; Sadraei et al., 2007 [332]) Hence, there is a need for methods which provide the stakeholder view as well as the problem view and which explicitly describe the collaboration between stakeholders and requirements engineers (Coughlan and Macredie, 2002 [105]; Nuseibeh and Easterbrook, 2000 [290]; Fuentes-Fernandez et al., 2010 [153]). The methods we have chosen are sufficient to reflect all these needs, even though there might be other methods which also allow to produce sufficient results using our process.

1.5. Solutions, Structure, and Conclusion

In this section, we give an overview of the papers we published and which are of relevance for this work. Additionally, we outline the structure of the rest of this work, and relate the structure to the proposed process and our published works.

Table 1.1 shows the papers we have published and which are related to this work. The first column of the table shows the *citation* of the paper, which is used in the following chapters. The second column indicates the *contribution* type. **Main** indicates that the author of this thesis is also the main author of the publication at hand, which means that he/she contributed more than the other authors. *Major* indicates that the author of this thesis is not the main author of the paper, but contributed a significant part of the paper at hand. Minor indicates that the

Citation	Contribution	Title	Chapters	Venue	Kind	Base
Beckers, Schmidt, Küster, and Faßbender (2011 [39])	Minor	Pattern-Based Support for Context Establishment and Asset Identification of the ISO 27000 in the Field of Cloud Computing	6, 22	ARES	Conf.	X
Beckers, Faßbender, Küster, and Schmidt (2012 [43])	**Main**	A Pattern-Based Method for Identifying and Analyzing Laws	8, 14, 15	REFSQ	Conf.	✓
Beckers, Faßbender, Heisel, and Meis (2012 [42])	Minor	A Problem-based Approach for Computer Aided Privacy Threat Identification	13	APF	Conf.	X
Beckers, Faßbender, and Schmidt (2012 [44])	**Main**	An Integrated Method for Pattern-Based Elicitation of Legal Requirements Applied to a Cloud Computing Example	8, 14 16	RISI (ARES)	Works.	✓
Beckers, Eicker, Faßbender, Heisel, Schmidt, and Schwittek (2012 [40])	**Main**	Ontology-Based Identification of Research Gaps and Immature Research Areas	3	CD-ARES	Conf.	✓
Beckers, Faßbender, Heisel, and Meis (2012 [41])	**Main**	Pattern-Based Context Establishment for Service-Oriented Architectures	5, 8, 10	Book	Chapter	✓
Beckers and Faßbender (2012 [38])	Minor	Peer-to-Peer Driven Software Engineering Considering Security, Reliability, and Performance	6	RISI (ARES)	Works.	X
Beckers, Faßbender, and Heisel (2013 [47])	**Main**	A Meta-Model for Context-Patterns	7, 8	EuroPLoP	Conference	✓
Beckers, Côté, Faßbender, Heisel, and Hofbauer (2013 [45])	*Major*	A pattern-based method for establishing a cloud-specific information security management system	6, 14, 15	RE	Journal	✓
Beckers, Faßbender, Heisel, and Paci (2013 [48])	*Major*	Combining Goal-oriented and Problem-oriented Requirements Engineering Methods	11, 18	CD-ARES	Conf.	X
Beckers, Côté, Hatebur, Faßbender, and Heisel (2013 [46])	Minor	Common Criteria CompliAnt Software Development (CC-CASD)	22	SAC	Conf.	X
Faßbender and Heisel (2013 [133])	**Main**	From Problems to Laws in Requirements Engineering Using Model-Transformation (Best Students Paper Award)	8, 14,15, 16, 17	ICSOFT	Conf.	✓
Faßbender and Heisel (2014 [134])	**Main**	A Computer Aided Process From Problems to Laws in Requirements Engineering	8, 14, 15, 16, 17, 21	Soft. Tech.	Journal	✓
Beckers, Faßbender, and Heisel (2014 [49])	**Main**	A Meta-Pattern and Pattern Form For Context-Patterns	7, 8	EuroPLoP	Conf.	✓
Beckers, Faßbender, Heisel, and Suppan (2014 [51])	Minor	A Threat Analysis Methodology for Smart Home Scenarios	8, 13, 22	Smart Grid Sec. (ESSOS)	Works.	X
Faßbender, Heisel, and Meis (2014 [136])	*Major*	Aspect-oriented Requirements Engineering with Problem Frames	12, 13	ICSOFT	Conf.	X
Alebrahim, Choppy, Faßbender, and Heisel (2014 [6])	**Main**	Optimizing Functional and Quality Requirements According to Stakeholders' Goals	11, 18, 19, 20	SQSA	Chapter	✓
Beckers, Faßbender, and Heisel (2014 [50])	**Main**	Deriving a Pattern Language Syntax for Context-Patterns	8, 9	EuroPLoP	Conf.	✓
Faßbender, Heisel, and Meis (2014 [135])	**Main**	Functional Requirements Under Security PresSuRE (Best Students Paper Award)	13	ICSOFT	Conf.	✓
Alebrahim, Faßbender, Heisel, and Meis (2014 [8])	*Major*	Problem-Based Requirements Interaction Analysis	18	REFSQ	Conf.	X
Alebrahim, Faßbender, Filipczyk, Goedicke, Heisel, and Konersmann (2014 [7])	**Main**	Towards a Computer-Aided Problem-Oriented Variability Requirements Engineering Method	18	ASDENCA (CAiSE)	Works.	X
Alebrahim, Hatebur, Faßbender, Goeke, and Côté (2015 [11])	Minor	A Pattern-Based and Tool-Supported Risk Analysis Method Compliant to ISO 27001 for Cloud Systems	22	Secur. Softw. Eng	Journal	X
Faßbender, Heisel, and Meis (2015 [138])	*Major*	A Problem-, Quality-, and Aspect-Oriented Requirements Engineering Method	12	Soft. Tech.	Journal	X
Faßbender, Heisel, and Meis (2015 [137])	**Main**	Problem-based Security Requirements Elicitation and Refinement with PresSuRE	13	Soft. Tech.	Journal	✓
Alebrahim, Faßbender, Filipczyk, Goedicke, and Heisel (2015 [10])	**Main**	Relating Performance and Security Tactics to Architectural Patterns	22	EuroPLoP	Conf.	X
Alebrahim, Faßbender, Filipczyk, Goedicke, and Heisel (2015 [9])	**Main**	Towards Systematic Selection of Architectural Patterns with Respect to Quality Requirements	22	EuroPLoP	Conf.	X
Faßbender and Aysolmaz (2015 [132])	**Main**	Transforming Process Models to Problem Frames	11, 12	IWPE (BPM)	Works.	✓

Table 1.1.: *Own Papers*

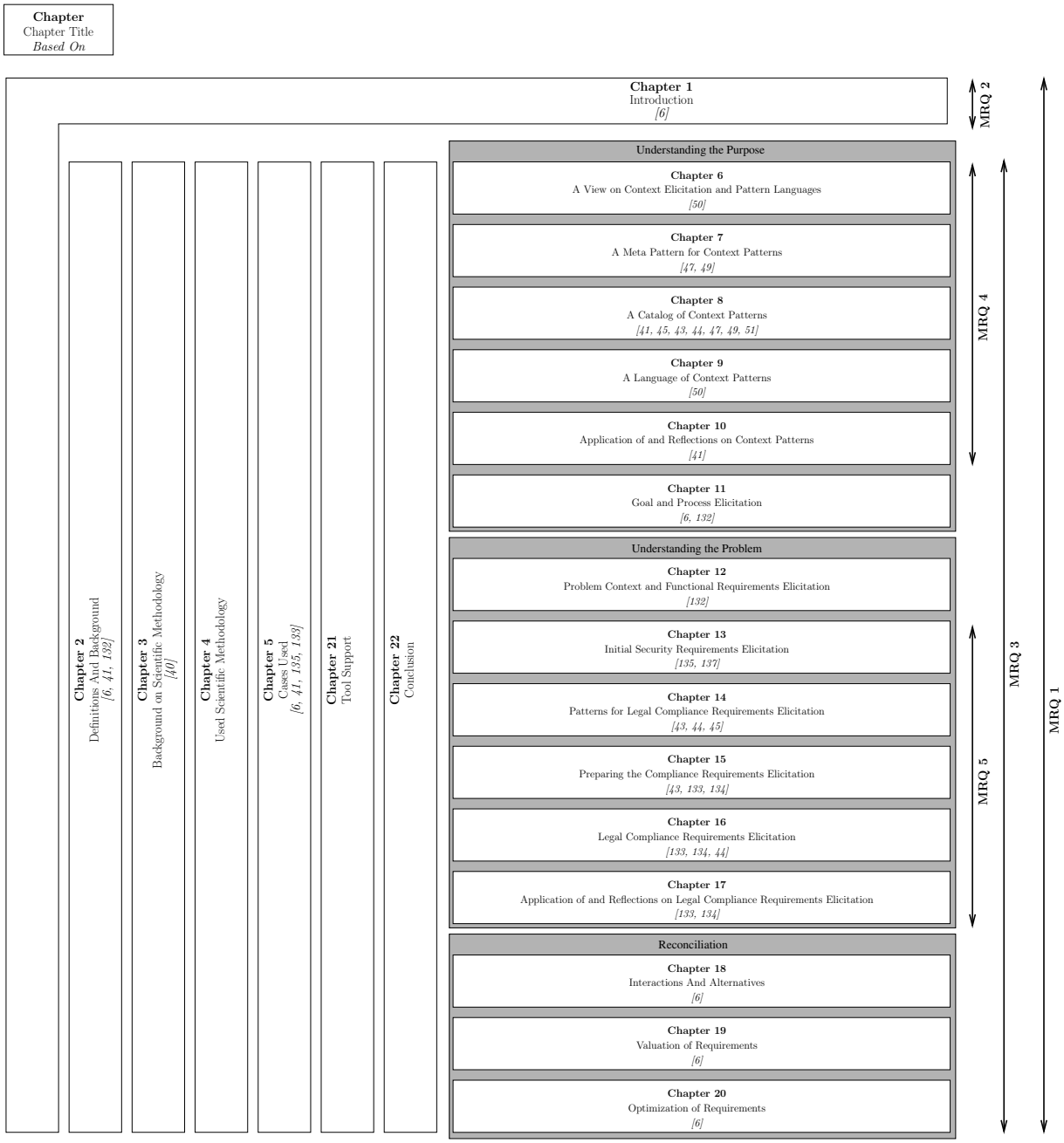


Figure 1.7.: *Overall Structure of the Thesis*

author of this thesis was involved in writing the paper, but contributed just a small part of the content. The *title* of the paper at hand is given in the third column, followed by the *chapters* the paper is relevant for. The fourth and fifth column show where the paper at hand was published and which kind of publication it is: conference paper (*Conf.*), workshop paper (*Works.*), journal paper (*Journal*), and book chapter (*Chapter*)). The last column indicates whether at least one chapter is based on (*Base*) the paper at hand (✓), or if this paper is just mentioned but not explained in detail (*X*). All in all, 27 own publications are used within this thesis.

The structure of the rest of this thesis is shown in Figure 1.7 and which MRQ is covered by which chapter. White boxes are used to visualize the chapters. For each chapter a box shows the chapter number, the chapter title, and the papers the chapter is based on.

Boxes with a vertical orientation indicate that this chapter deals with crosscutting topics. Such crosscutting chapters are the introduction (Chapter 1), the background in which important notations and terms are defines (Chapter 2), an introduction in the general scientific background relevant for this thesis (Chapter 3), the actually used scientific methods (Chapter 4), the introduction of the running example and important cases used in later chapters (Chapter 5), Chapter 21 which outlines the tool support which was developed to support the process and activities, and finally the chapter which concludes this thesis (Chapter 22).

The chapters for which the boxes have a horizontal orientation follow the process as described in Figure 1.6. The activity *context elicitation* is covered by Chapter 6 which introduces the matter of context elicitation and patterns, Chapter 7 which is concerned with the Meta Pattern for context elicitation, Chapter 8 introduces a catalog of context patterns, Chapter 9 the according pattern language, and Chapter 10 shows the application of a context pattern to the running example and discusses the validation for our context patterns. The activities *goal elicitation* and *process elicitation* are covered by Chapter 11 in which we use existing methods and notations to model the goals and processes. In Chapter 12 we show how to turn the processes into requirements and model the context of the system-to-be and the requirements accordingly (Activities *problem context elicitation* and *functional requirements elicitation*). For the activity *quality requirements elicitation* we go into detail on the matter of security and how to refine a generic security goal into concrete security requirements in Chapter 13. Chapter 14 introduces the topic of legal compliance requirements engineering and patterns for this matter, Chapter 15 explains how to prepare the actual elicitation of legal compliance requirements, Chapter 16 shows the actual elicitation, and Chapter 17 discusses the sufficiency of the proposed solution for legal compliance requirements elicitation. These chapters cover the activity *compliance requirements elicitation*. Chapter 18, Chapter 19, and Chapter 20 cover the activities *detection of interactions*, *generation of alternatives*, *valuation of requirements*, and *optimization of requirements* respectively.

In the following we sum up this chapter:

- introduced the matter of requirements engineering in general and the importance of decision making and requirements selection in particular,
- introduced the research questions considered within this thesis,
- taken a closer look at decision making in RE, which information is necessary for an informed requirements selection, and which challenges one has to consider when forming a requirements selection solution,
- outlined a general process to collect the important information and support the decision making,
- introduced the papers we have published which are of relevance for this thesis,
- and outlined the following chapters.

Chapter 2

Definitions And Background

In this chapter we introduce important terms (Section 2.1) as well as methods and notations used within this work (Section 2.2).

2.1. Important Terms

In this section we introduce terms specific to requirements engineering (Section 2.1.1), specific to the field of legal compliance (Section 2.1.2), and some further terms (Section 2.1.3).

2.1.1. Requirements Engineering Terms

The following requirements engineering (RE) specific terms are of importance for understanding the RE related content of this work. The definitions are based on Jackson (2001 [200]), and Zave and Jackson (1997 [399]).

Environment The environment is the part of the real world relevant to our problem we want to solve by developing a particular software.

Domain Domains are entities in the environment, such as sensors, actuators, people, and so forth.

Machine The machine is the particular domain to be built by us. It realizes the desired properties of the system.

System A system consists of the environment and the machine.

Phenomena States and behavior of the environment are described by phenomena. A phenomenon can be, for example, an event, action or operation (occurring in the environment). Domains can observe or control phenomena that occur at the interfaces directly connected to it.

Requirement A requirement R is an optative statement expressing a desired behavior of the environment once the machine is in operation. Accordingly, requirements do not refer to the machine but only to the environment. Hence, they describe the characteristics of the system which are currently (without the machine) deficient or not even there, and which shall be assured by the machine.

Domain Knowledge The domain knowledge is described by indicative statements about the environment which apply with and without the existence of the machine. Domain knowledge consists of assumptions, facts, and definitions or designations.

Assumptions Assumptions describe conditions that are needed in order to enable the fulfillment of the requirements. But assumptions are not always true. Hence, each environment has to be checked whether the assumptions are met or not. Usually, assumptions describe required user behavior or other assumed properties of a domain.

Facts In contrast to assumptions, facts are true in all cases. Facts are, for example, physical laws of the behavior of systems in the environment, but also a specification of an external system.

Designation "A designation is an informal description of the meaning of an atomic term referring to the environment." (Zave and Jackson, 1997 [399])

Definition "A definition is a formal description of the meaning of an atomic term, using other definitions or designated terms." (Zave and Jackson, 1997 [399])

Specification While requirements describe how the environment should behave once the machine is integrated into it, a specification describes the machine behavior itself in a detailed way. The machine is constructed in accordance to the specification afterward. Specifications are implementable requirements. They are derived from the requirements using domain knowledge.

2.1.2. Legal Compliance Terms

The Black legal dictionary (Black and Garner, 1999 [58]) provides the following definitions for the used terms from the legal domain.

Compliance is the adherence to established laws and regulations (Black and Garner, 1999 [58]). A compliant software system is a system that fully adheres to its specifications, including legal aspects.

Law is commonly referred to as the set of enforced rules that govern social institutions by defining their rights and obligations, as well as penalties when the latter are violated (Black and Garner, 1999 [58]).

Regulation is the act or process of controlling by rules or restrictions (Black and Garner, 1999 [58]).

Statute is a formal written enactment of a legislative authority that governs a state or a country (Black and Garner, 1999 [58]). Typically statutes command or prohibit something and are often used to distinguish law made by legislative bodies from cases, decided by courts.

Case is a legal dispute between opposing parties resolved by court or an equivalent legal process. A legal case might be civil or criminal proceeding and always includes a defendant and an accuser.(Black and Garner, 1999 [58])

Within this thesis we will use the term **legal compliance** instead of the more general term compliance as defined by Black and Garner (1999 [58]).

> "Legal [compliance] refers to an organization's ability to maintain a defensible position in a court of law." (Breaux, 2010 [64])

This definition differs in two central aspects from the one of Black and Garner (1999 [58]). First of all, within legal compliance we are focusing only on those regulations of importance at a court, namely, laws and cases. Moreover, it is not only of relevance that an organization in general

or a system in specific adheres to the relevant laws, but also that one can prove adherence and convince the judges at a court.

Beside these definitions there are some particularities of legal systems the reader should be aware of. First of all, there is the distinction between so called **statute law** and **case law** (Apple and Deyling, 1995 [18]). Within statute law, a judge has always to consider and judge based on the statues enacted by the legislator. Former cases are only treated as inspiration, and are not binding in any case. Hence, in statute law, the statutes are the central means for judging cases. In case law the situation is just the other way round. Statutes form only the baseline for judging a case. A judge might always extend or modify the implications of a statute whenever the case at hand does not directly meet the circumstances described by a statute. However, former cases which are applicable to the case at hand are binding. Hence, over time the cases become the means which drive judgments, not the statutes. This is a fundamental difference between statute law and case law one has to keep in mind when investigating legal compliance.

A further difference between law systems is about the scope of laws, and how central concerns are treated (Apple and Deyling, 1995 [18]). In an **omnibus law system** central concerns which are not specific to a certain scope are regulated using laws which cover all aspects of life within a jurisdiction. For example, the federal data protection act of Germany (Bundestag der Bundesrepublik Deutschland (Lower House of German Parliament), 2009 [88]), which regulates the privacy concern, is valid for all kinds of industries, situations, and so forth. In contrast, within a **sectoral law system** even central concerns such as data protection are regulated for each sector differently. For example, in the US laws are organized according to different industries, and for each industry a specific data protection act exists.

A last difference is about the way the laws are formulated (Apple and Deyling, 1995 [18]). A **prescriptive law** (or best practice law) not only describes when to apply the law and what the expected outcome of being compliant is, but also how to achieve compliance. An **outcome-based law** does only describe when the law is applicable, and what the expected outcome of being compliant is. How to achieve compliance is not described by such a law.

2.1.3. Further Terms

Further terms we would like to introduce are stakeholder (Section 2.1.4), context (Section 2.1.4), and service oriented architecture (Section 2.1.6). The latter definition is based on Beckers, Faßbender, Heisel, and Meis (2012 [41])

2.1.4. Stakeholder

A general definition for stakeholder from the view of an organization is given by Freeman (2010 [152]):

> "A stakeholder ... is (by definition) any group or individual who can affect or is affected by the achievement of the organization's objectives."

A more requirements engineering specific definition is given by Sommerville and Kotonya (1998 [359]):

> "System stakeholders are people or organizations who will be affected by the system and who have a direct or indirect influence on the system requirements"

Based on these two definitions, we derive the following definition: A **stakeholder** is any group or individual who is indirectly or directly influenced by the machine, who is indirectly or directly affected by a changed behavior of an organization due to the machine, who indirectly or directly influences the machine when running, or who can indirectly or directly influence the requirements for this machine.

Note that this definition partitions the stakeholders into **direct stakeholders** which correspond to domains in the **direct environment** of a machine as they directly interact with or

are directly influenced by the machine, whereas there are also **indirect stakeholders** which do not directly interact with or are not directly influenced by the machine. We consider these stakeholders to be part of the **indirect environment**.

2.1.5. Context

The term context is widely used in different scientific areas, and also in different software engineering topics (Brézillon, 1999 [74]). In consequence, many different definitions exist. For example, for context aware systems a common definition is

> "Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant for to the interaction between a user and an application, including the user and application themselves"(Dey, 2001 [117])

Such a definition is sufficient to describe the direct environment of a machine and the information necessary to be observed by the machine to react according to the context. But as we have already seen for the stakeholders, there is also an indirect environment including, for example, further stakeholders. Additionally, for requirements engineering, we need further knowledge beside the current characteristics of a situation. Hence, we define **context** as: Every entity which is directly or indirectly influencing the requirements for a machine or later on directly or indirectly influencing the behavior of the machine when running is part of the context of this particular system. The context is defined by the knowledge about these entities. An entity is a person, existing system, or any other object considered to be relevant.

2.1.6. Service Oriented Architecture

Various definitions of the term **Service Oriented Architecture (SOA)** exist, because the SOA concept spans a wide field of research areas and technologies. However, there is a common understanding about some core characteristics of SOA. First, a SOA is modular with a high autonomy of its parts, not only in the sense of interaction within the architecture, but also in the sense of , for example, autonomous stakeholders and development teams (Dijkman and Dumas, 2004 [119]; Papazoglou, Traverso, Dustdar, and Leymann, 2008 [306]; Perepletchikov, Ryan, Frampton, and Schmidt, 2008 [309]). Second, services have a coarse granularity, encapsulating more or less complex tasks. As a result, a single service is a complex product (Dijkman and Dumas, 2004 [119]). Third, SOA is process-driven (Dijkman and Dumas, 2004 [119]). In most cases, a service performs one activity of a business process (Arsanjani, Ghosh, Allam, Abdollah, Gariapathy, and Holley, 2008 [20]). Hence, a SOA has to be designed to fulfill business requirements and goals (Perepletchikov et al., 2008 [309]). Fourth, the services of a SOA have to be loosely coupled (Papazoglou et al., 2008 [306]). In practice, the business processes to be supported by a SOA change frequently. In consequence, a SOA has to evolve dynamically (Papazoglou et al., 2008 [306]). Hence, the services are loosely coupled to enable dynamic (re)assembly. Fifth, the re-usability of services is high (Perepletchikov et al., 2008 [309]; Papazoglou et al., 2008 [306]; Arsanjani et al., 2008 [20]). The re-usability is a result of the autonomy of services and the loose coupling between them. Sixth, a SOA is a distributed system (Papazoglou et al., 2008 [306]; Dijkman and Dumas, 2004 [119]). The reasons are that business processes cross the border of one enterprise (Rodríguez, Fernández-Medina, and Piattini, 2007 [323]) and that services can be offered by third-party service providers (Papazoglou et al., 2008 [306]). Seventh, SOA is technology-independent (Papazoglou et al., 2008 [306]; Dijkman and Dumas, 2004 [119]). SOA is meant to integrate highly heterogeneous services, which means that the technologies used to implement the services can differ. Summarizing, SOA can be characterized as a process driven,

modular, technology-independent, dynamic and distributed system, which relies on reusable, autonomous, loosely coupled and coarse-grained services.

2.2. Methods and Notations Used

In the following, we introduce the methods and notation used within this thesis. We use agendas (Section 2.2.1) for describing our methods, $i*$ for expressing goals (Section 2.2.2), UPROM for modeling business processes (Section 2.2.3), problem frames for capturing requirements (Section 2.2.4), the analytical network process for valuating requirements (Section 2.2.5[1]), and optimization for selecting the final set of requirements (Section 2.2.6[2]). Beside Section 2.2.1, which is new content, all other sections are based on (Faßbender and Aysolmaz, 2015 [132]) and Alebrahim, Choppy, Faßbender, and Heisel (2014 [6]).

2.2.1. Agenda Concept and Notation

Heisel (1998 [178]) introduced the concept of an agenda, which contains a list of phases and steps to be performed when carrying out some task in the context of software engineering activities. For each step, it is explicitly stated which inputs are required to conduct the step. Each step results in one or more documents that are expressed in a certain language, for example, natural language, graphical representations, models such as UML (Unified Modeling Language ("UML Revision Task Force", 2009)) models, formal languages, and so forth. For each step, agendas contain an informal description of the step, which may depend on other steps. Agendas are a means to guide systems and software development activities. Additionally, agendas enable quality assurance, because to each step validation conditions are associated that help to detect errors as early as possible in the process.

To visualize an agenda we have developed a graphical notation which we will use in the following chapters. Figure 2.1 shows the elements of this notation. An agenda is structured into 5 *lanes*. *Lane 3* is the *process* lane which is showing the *phases* and *steps* of the agenda. Phases are depicted as rounded rectangles with bold font, and steps are depicted as rounded rectangles

[1] Page 37
[2] Page 39

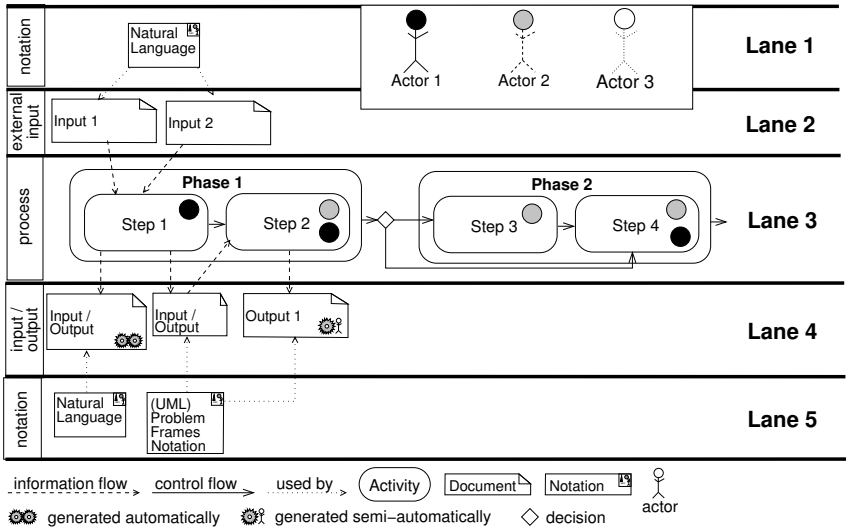


Figure 2.1.: *Agenda Notation*

with normal font. Both are also referred to as *activities*. Phases contain steps. Phases and steps are connected using solid arrows which indicate *control flows*. A control flow defines the possible sequences of activities. There might be conditional control flows. The *decisions* taken for these flows are visualized by diamonds. Another important element are the *actors* depicted as stick-figures. They are the ones conducting certain steps. Each actor has a head of different color. Whenever a head of a particular color is attached to an activity, it means the according actor is involved into this activity.

Above the process lane there is the *external input lane*. This lane shows all input which is not created within the process, but which is necessary to conduct at least one step. Inputs are referred to as *documents* which are depicted as notes (rectangular with a dog-ear). The *information flow* from a document to an activity is visualized using a dashed arrow. Below the process lane there is the *input / output* lane. Here, the documents generated by the process are shown. A document generated by one step might be used in further steps. Documents in the input / output lane might be annotated with two gear wheels, or a gear wheel and a stick figure. This indicates that the document is generated fully automatically or semi-automatically respectively.

Last, there are two outer lanes (Lane 1 and 5). These lanes are the *notation lanes* and show the notation *used for* expressing a document. A notation is linked to a document using a dotted arrow. A notation is depicted as rectangular with some small pictures in the upper right corner.

2.2.2. i* Goal Notation

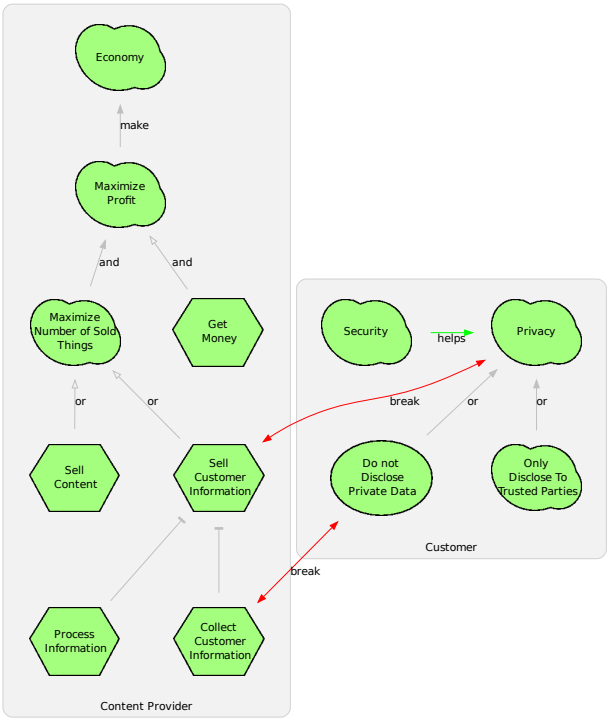


Figure 2.2.: *Example Goal Tree*

The $i*$ framework was developed for modeling and reasoning about organizational environments and their information systems. It consists of two main diagram types (Yu, 1996 [396]). The Strategic Dependency (SD) model is used to describe the dependency relationships among various actors in an organizational context. The Strategic Rationale (SR) model is used to describe stakeholder interests and concerns (Yu, 1997 [397]). For our work we used the strategic rationale model, because the relations between stakeholders are not of interest for us but the goals a stakeholder pursues. The following brief introduction is based on (Yu, 1996 [396]; Yu, 1997 [397]), and the official $i*$-wiki[3]

In the $i*$ framework, the Strategic Rationale model provides a more detailed level of modeling by looking "inside" actors to model internal intentional relationships. Figure 2.2 shows an example Strategic Rationale model for a scenario in which a customer wants to buy some content from a content provider. The goal graph shown is incomplete and only contains such elements needed to explain the $i*$ notation.

[3]http://istar.rwth-aachen.de/tiki-index.php?page=i*+Guide

The first element of a goal graph is the actor visualized by the actor boundaries (see Figure 2.2: light gray rectangles with rounded corners). All of the elements within a boundary for an actor are explicitly desired by that actor. For our example, the customer and the content provider are actors. A goal of an actor represents an intentional desire of this actor. For goals $i*$ distinguishes between hard goals, soft goals and tasks. A hard goal defines a desire for which the satisfaction criteria are clear, but the way of satisfying it is unspecified. Hard goals are visualized as ellipses (see Figure 2.2). An example is the hard goal *do not disclose private data* of the customer. It is satisfied whenever the customer does not disclose private data, which is measurable, but is unspecific about the process how to avoid the disclosure. A soft goal is even more underspecified, as for a soft goal even no clear satisfaction criteria are known. Soft goals are denoted as clouds (see Figure 2.2). An example is the high-level goal "Maximize Profit", because at this level one cannot precisely state what the maximum profit would be or a process to achieve it. In contrast, a task defines both, the criteria for fulfilling the goal and the process to do so. Tasks are denoted as hexagons (see Figure 2.2). For example, the content provider has already a process for *process information* and it is clear when the processing is successful.

Goals are connected by links. The first kind of link is the contribution link. Contribution links are visualized using arrows with filled arrowheads (see Figure 2.2). A contribution link between a child goal (tail of the arrow) and a parent goal (arrow head) means that the child goal contributes to the satisfaction of the parent goal. The second kind of link is the means end link, which is visualized using a unfilled arrowhead. It connects tasks (tail of the arrow) with (soft)goals (arrow head) and indicates that this process is a necessary means to satisfy a (soft)goal. The annotation of these arrows specifies the kind of contribution. A *break* denotes that the child denies the parent. For example, when the content provider *sells the customer information*, it breaks the *privacy* goal of the customer. A *make* denotes that if the child is satisfied the parent is satisfied, too. For example, in case the content provider successfully *maximizes its profit*, its high level economic goal is also met. An *or* means that at least one of the children has to be satisfied for the satisfaction of the parent. For example, for *maximizing the number of sold things*, the content provider has to *sell content*, *sell customer information*, or both. For an *and* contribution all children have to be satisfied. For example, to *maximize the profit*, the content provider has to *maximize the number of sold things* as well as the provider has to assure that it *gets the money*. *Hurts* specifies a negative influence, which does not necessarily break the parent goal. *Helps* is used whenever the child goal has a positive influence, but is not necessarily needed to fulfill the parent goal. For example, *security* helps to maintain *privacy* as attackers might not be able to collect private data, but it is not sufficient as the content provider can still collect and sell the data. The last kind of link is the decomposition link. It is used to decompose a goal to more fine-grained parts. A decomposition link is denoted as arrow with a T-head (see Figure 2.2). For example, the process of *selling customer data* includes the *collecting* and *processing* of the customer data.

2.2.3. UPROM

In UPROM as proposed by (Aysolmaz, 2014 [23]), business processes are analyzed by developing extended event-driven process chains (EPC), value chain diagrams, function trees and organization chart diagrams. EPC, the core process modeling diagram for UPROM, is a popular process analysis notation introduced by the ARIS framework (Scheer, 1998 [339]). ARIS is a popular and leading business process management tool in the industry (Norton, Blechar, and Jones, 2010 [288])

An EPC diagram consists of control flow elements, organizational elements, information artifacts, applications and business rules. Figure 2.3 shows a small example including all elements. Control flow elements are events (pink hexagon), such as *payment canceled*, functions, such as

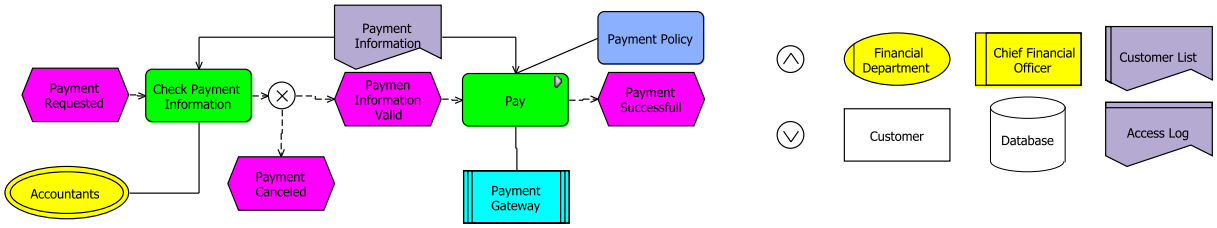


Figure 2.3.: *EPC Example*

check payment information, which are also referred to as activities, (green rectangle with rounded corners) and gateways (white circles). There are gateways to express XOR (white circle with an X), AND (white circle with an $\wedge$), and OR (white circle with an $\vee$) decisions. They are used to model different control flows which define the possible sequences of tasks in a process. For example, in Figure 2.3 there is the decision whether the *check payment information* function was successful ending in an *payment information valid* event, or if it was not successful and the *payment is canceled*.

Organizational element types are organizational unit (yellow ellipse with a vertical stripe on the left), such as a *financial department* within a company, group (yellow double ellipse), such as the *accountants* within this department, position (yellow rectangle with a vertical stripe on the left), such as the *chief financial officer*, and external person (white rectangle), such as the *customers*. They are connected directly to the functions to indicate the responsibility for, or the involvement in, the related function. In Figure 2.3, the *accountants* are involved in the function *check payment information*. Organizational elements can also be connected to an information artifact to model that the organizational element provides the artifact or retrieves the artifact. Information artifacts are document (violet paper snippet), such as the form containing the *payment information*, list (violet paper snippet with a vertical stripe on the left), such as a *customer list*, log (violet paper snippet with a horizontal stripe on top), such as an *access log*, and file (white cylinder), which can be any type of data storage such as a *database*. These elements provide different visualizations to represent information used or created by the process. The application element (turquoise rectangle with double vertical stripes on both sides), such as the *payment gateway*, is connected to the function to model the system which supports the function execution. A business rule (light-blue rounded rectangle), such as the *Payment Policy*, is connected to the function to identify the constraints to be considered for the execution of the function. An EPC can also contain invocations of other processes. A sub-process, which is too complex to include it in the current EPC, is depicted as normal function which is annotated with a triangle (*Pay*). Each function which does not represent a sub-process is also called leaf function.

To move on to requirements analysis based on business processes, each leaf function in EPC diagrams is analyzed for automation potential. If the function is to be (semi-) automated, a functional analysis diagram (FAD) is created as a sub-diagram. A FAD serves the purpose of analyzing how the function will be executed by identifying the responsibilities, related entities, operations on entities and constraints. Figure 2.4 shows an example for an FAD, in which an

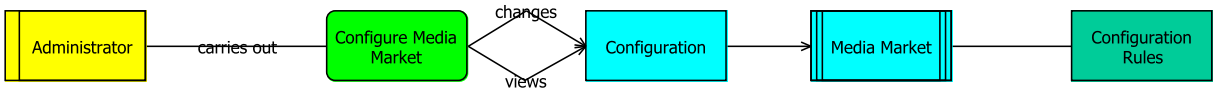


Figure 2.4.: *FAD Example*

application called *Media Market* is supposed to be configured by an *administrator*. An FAD includes elements of the types function, entity, application, organizational element (position, organizational unit, external person), and constraint. The function element (*configure media market*), which corresponds to exactly one function in an EPC diagram, beeing refined by the FAD, is in the center of the FAD. Organizational elements, in our case the role *administrator*, are connected to the function via a connection which is tagged with an involvement type. Involvement types are "carries out", "approves", "supports", "contributes to", "must be informed on completion". In this way, the responsibilities to conduct the function in a process aware information system (PAIS) are identified. Entities (turquoise rectangle), which are required for or during the execution of the function, are connected to the function. An entity represents any kind of information artifact, in our case the actual *configuration*. The operations executed on these entities are defined by the connection name. The possible operations are uses, views, creates, changes, reads, deletes, and lists. For our example, the *configuration* is *viewed* and *changed*. Each entity or cluster is also connected to the application on which it resides. In our case, the application is the *media market*. If there are further constraints that restrict how the system operates, they are modeled as constraints and connected to the related application, such as the *configuration rules* in our FAD depicted in Figure 2.4. During development of FADs, conceptual definitions of entities are revealed.

Entity relationship (ER) diagrams are modeled and used to grasp a full view of these entities with aggregation and generalization relations between them. Figure 2.5 shows an example diagram. The entity *Participant* has the two attributes (gray ovals) name and age. It is the supertype (green triangular) of *customer* and *employee*. Participants are related to (yellow diamond) the entity *company celebration*, because they are invited to this celebration. Integral part of the celebration is the *program* and the *menu* for the evening.

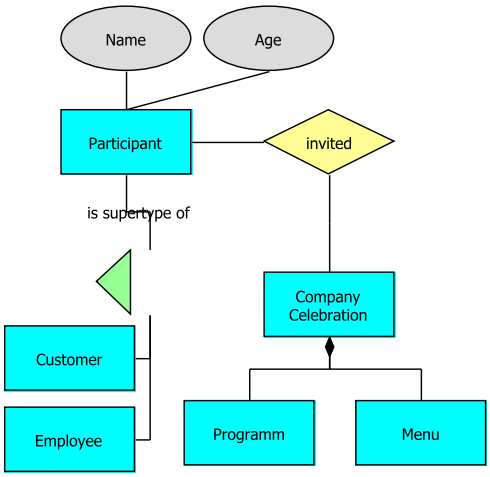


Figure 2.5.: *ERD Example*

2.2.4. Problem Frames

Note that we use a UML-based adaption of problem frames, which is implemented as a specific UML profile for problem frames (UML4PF) as proposed by Hatebur and Heisel (2010 [176]). Hence, the graphical representation differs from the original representation proposed by Jackson (2001 [200]), but the semantics remain the same. To be able to annotate problem diagrams with quality requirements, Alebrahim, Hatebur, and Heisel (2011 [4]) extended the problem frames notation. This enables us to complement functional requirements with quality requirements.

The objective of requirements engineering in terms of Jackson (2001 [200]) is to construct a *machine* (i.e., system-to-be) that controls the behavior of the environment (in which it is integrated) in accordance with its requirements. The first step towards understanding and defining the problem to be solved by the machine is to understand the context of the machine. The context of the machine is given by the environment in which the problem to be solved is located, and in which the machine will be integrated to solve the problem. The environment is defined by means of domains and interfaces between these domains and the machine. The information about the machine and its environment is modeled in a so called *context diagram*. Figure 2.6 shows a context diagram for a small patient monitoring system which tracks the vital

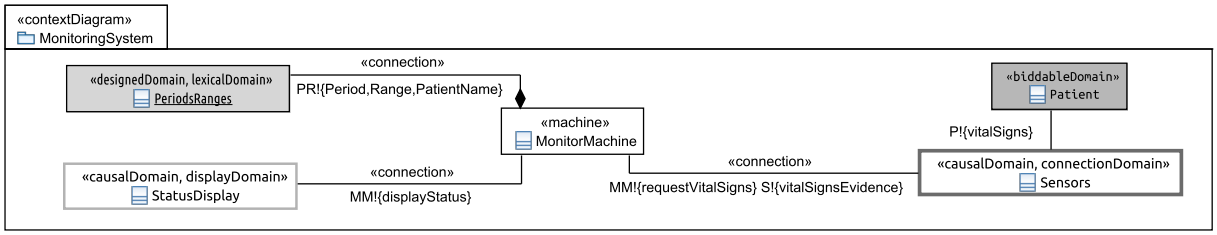


Figure 2.6.: *Context Diagram for a Patient Monitoring System (based on Jackson (2001 [200]))*

signs of a patient and displays it accordingly. Jackson distinguishes the domain types *machine* (class with the stereotype ≪`machine`≫) which is the thing to be built, *biddable domains* (classes with the stereotype ≪`biddableDomain`≫) that are usually people, *causal domains* (classes with the stereotype ≪`causalDomain`≫) that comply with some physical laws or the specification is known, and *lexical domains* (classes with the stereotype ≪`lexicalDomain`≫) which represent data. Additionally, a domain can be *designed* (≪`designedDomain`≫), which means we can influence its structure and behavior, a *connection* (≪`connectionDomain`≫), which means the domain establishes the communication between two other domains, and / or a *display*, which means it is used to display information. For our example, the *MonitorMachine* is our machine. It obtains the vital signs of the *Patient* who is a biddable domain using several *Sensors* which are causal domains and, as they establish the link between the machine and the patient, connection domains. The obtained data is stored in the *PeriodRanges* which is a lexical domain and a designed domain as we decide how the data representation looks like. Finally, there is the *StatusDisplay* which is a causal and display domain. In the Problem Frames notation, *interfaces* connect domains and they contain *shared phenomena*. Shared phenomena may, for example, be events, operation calls or messages. They are observable by at least two domains, but controlled by only one domain, as indicated by "!". In Figure 2.6 the notation $MM!\{displayStatus\}$ (between the machine domain *MonitorMachine* and the causal domain *StatusDisplay*) means that the phenomenon *displayStatus* is controlled by the machine *MonitorMachine*. All other domains which are connected by the interface the phenomenon is part of can observe it. In our case, $MM!\{displayStatus\}$ is observed by the domain *StatusDisplay*.

Problem frames are a means to describe and classify simple software development problems. A problem frame is a kind of pattern representing a class of software problems. It is described by a frame diagram, which consists of *domains*, *interfaces* between them, and a *requirement* (class with stereotype ≪`requirement`≫). Figure 2.7 shows the problem frame for model building. The problem frame model building contains the machine **ModelBuilder** which is the software piece or system which shall later on fulfill the model building as described by the requirement

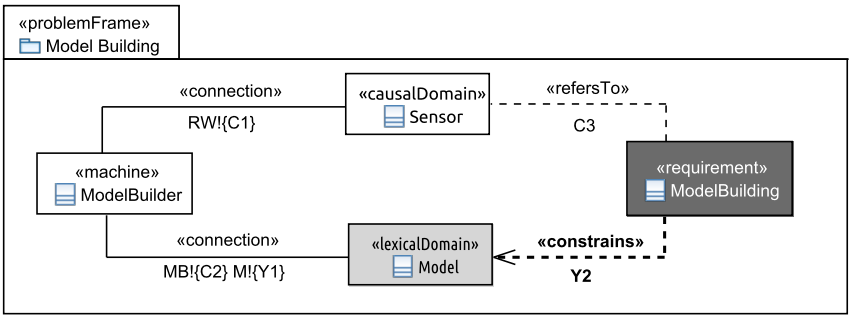


Figure 2.7.: *Problem Frame Model Building*

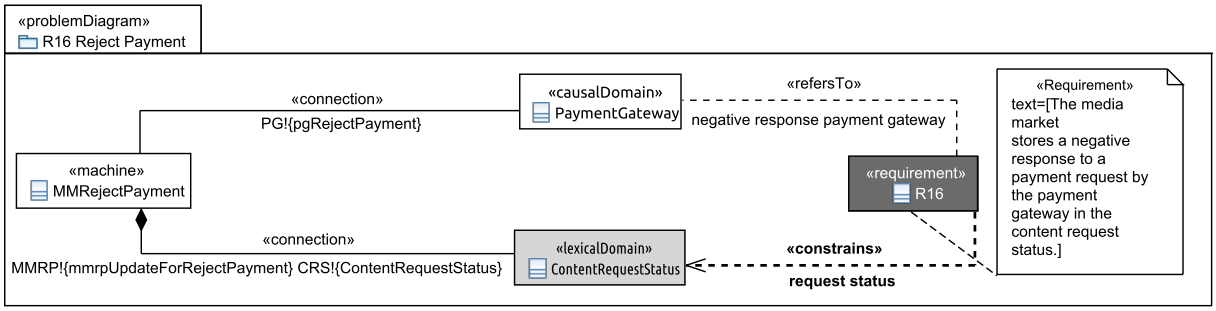


Figure 2.8.: *Problem diagram for R16 Reject Payment*

Model Building. The causal domain **Sensor** is used to observe the information about the real world from which the model is built. The lexical domain **Model** shall then reflect the result of the model building.

Requirements analysis with problem frames provides a decomposition of the overall problem into sub-problems, which are represented by *problem diagrams*. A simple problem diagram is an instance of a *problem frame*. Figure 2.8 shows a problem diagram which is about storing the response of a payment gateway in an according status. When we state a requirement we want to change something in the environment of the machine. Therefore, each requirement talks about and *constrains* at least one domain. These domains have to be influenced or changed to fulfill the requirement. For example, the domain **ContentRequestStatus** (class *ContentRequestStatus* with stereotype ≪`lexicalDomain`≫) is constrained by the requirement **R16** (class *R16* with stereotype ≪`requirement`≫) as shown in Figure 2.8 (dependency with stereotype ≪`constrains`≫ between class *R16* and class *ContentRequestStatus*), because to fulfill the requirement the **ContentRequestStatus** has to change accordingly to the rejection notification sent by the payment gateway. A requirement may also *refer to* several other domains in the environment of the machine which provide necessary information for fulfilling the requirement. The requirement **R16** refers to the domain **PaymentGateway** (class *PaymentGateway* with stereotype ≪`causalDomain`≫) (dependency with stereotype ≪`refersTo`≫ between class *R16* and class *PaymentGateway*, because the payment gateway sends the initial rejection notification).

The problem diagram shown in Figure 2.8 is an instance of the problem frame *Model Building* as it refers to the causal domain *PaymentGateway*, which instantiates the *Sensor*, and constrains the lexical domain *ContentRequestStatus*, which is an instantiation of *Model*.

2.2.5. Analytical Network Process

The *analytical network process (ANP)* is a generalization of the more widely known *analytical hierarchy process (AHP)* (Saaty, 2008 [328]; Saaty, 2005 [327]). Both rely on *goals*, *criteria* and *alternatives*. These *elements* are grouped by *clusters*, which can be ordered in *hierarchies* (AHP, ANP) or *networks* (ANP) (see Fig. 2.9). A goal in this context is the desired outcome of a decision process, for example, the “best system” or the “optimal marketing strategy”. A criterion is one important property, which has an influence on the decision regarding the goal. An alternative is one possible solution (part) to fulfill the goal and is compared to other alternatives with respect to the criteria. Hierarchy means that there is a strict order of influence between elements of different hierarchy levels, for example, sub-criteria are compared with respect to criteria but not the other way round. The top level of the hierarchy is formed by the elements which are not influenced by any other element. The bottom level form the elements which do not influence other elements. In contrast to AHP, which only allows influential relations between

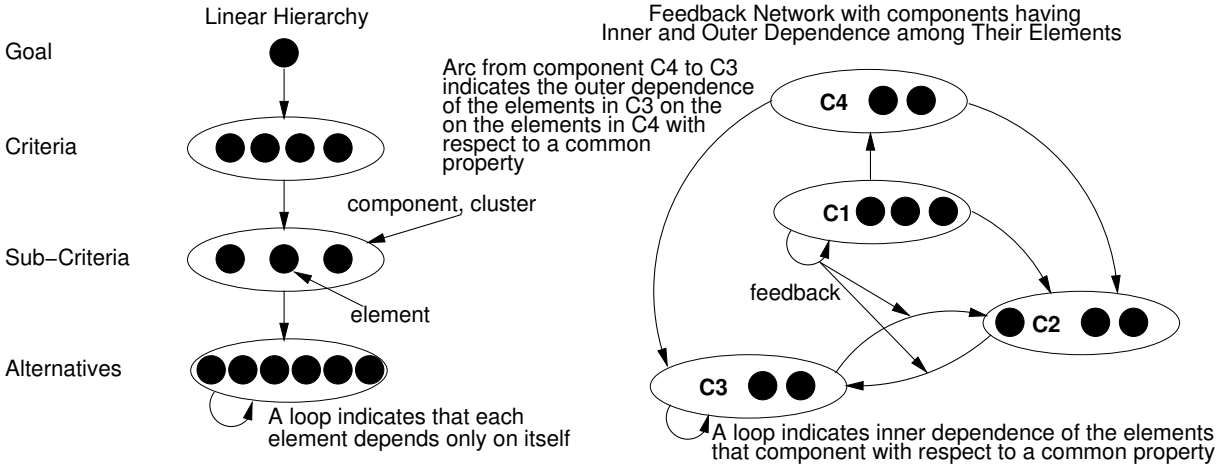


Figure 2.9.: *Hierarchy compared to a network [327]*

elements of adjacent hierarchy levels and only from the higher level to the lower one (see Fig. 2.9 on the left-hand side), ANP allows one to consider influential relations between elements within one level and bidirectional relations between all hierarchy levels forming a network (see Fig. 2.9 on the right-hand side). Note that ANP allows a mixture of hierarchy and (sub-)networks. Hence, ANP allows one to model more complex decision makings more accurately than AHP. The downside of ANP compared to AHP is the increasing number of comparisons to be made and the complex calculations to be executed (Saaty, 2005 [327]). On the one hand, ANP takes more time and the final decision is more difficult to understand, but on the other hand ANP allows a much deeper problem understanding and modeling and avoids errors due to over-simplification, which often occur when using AHP (Saaty, 2005 [327]; Saaty and Ozdemir, 2003 [330]). The steps of ANP are as follows (Saaty, 2008 [329]; Saaty, 2008 [328]; Saaty, 2005 [327]; Saaty and Ozdemir, 2003 [330]):

1. **Describe the decision problem** The first step of ANP is to understand the decision problem in terms of stakeholders, their objectives, the criteria and sub-criteria, alternatives, and the influential relations among all those elements. ANP gives no guidance for this step, but it is crucial for the success of the whole process.

2. **Set up control criteria** Besides the criteria and sub-criteria relevant for the decision, Saaty recommends to use control criteria for many decisions. He suggests benefits, opportunities, costs, and risks (BOCR) as control criteria (Saaty, 2008 [329]). Using control criteria allows one to model different dimensions of a decision problem and to combine negative and positive dimensions. Using control criteria is optional.

3. **Set up clusters** To structure the (sub-)criteria and alternatives and to make the network and the later comparisons better manageable, the (sub-)criteria and alternatives can be merged into clusters, for example, regarding their relation to a parent criterion. It is also allowed to have elements, which represent sub-networks to the network the element resides in, within a cluster. In this way very complex networks can be handled in a “divide and conquer” style.

4. **Relate elements of the network** The (sub-)criteria and alternatives have to be related according to their influence on each other. At this point the relation is undirected. Only the elements of one cluster are allowed to be directly related (inner dependence influence).

Clusters are related whenever at least one element of the first cluster is related to at least one element of the second cluster (outer dependence influence).

5. **Determine the influence direction** For each relation it has to be decided whether it is an unidirectional or bidirectional relation. One has also to decide whether a direction means "influences target element" or "is influenced by target element". The first option is recommended.

6. **Set up supermatrix** For each control criterion a supermatrix has to be constructed. Each element has to have a row and column representing it. Rows and columns are grouped according to the clusters. The cells of the supermatrix are marked whenever the element of the column influences the element of the row, or the cluster the column element belongs to influences the cluster of the row element.

7. **Compare elements** In this step, the pairwise comparison of elements, according to the inner and outer dependences of the cluster they belong to, has to be carried out. This results in an unweighted supermatrix.

8. **Compare clusters** To weight the different clusters, all clusters are compared pairwise with respect to a (control/ sub) criterion or goal. The resulting weights are then used to weight the cells of the columns, whose elements belong to the cluster. In this way, one obtains the weighted supermatrix.

9. **Compute limited supermatrix** The limited supermatrix is computed by raising the weighted supermatrix to a certain power k. The constant k can be freely chosen. For a low k the limited supermatrix might not be stable in the sense that for some elements, given by its row, the actual value does not converge to its final value. Hence, the priority of the element is not stable and cannot be determined. For a high k small priorities might drop to zero.

10. **Synthesize results to the control level** Set up a formula and weights for relating the control criteria. The result is the weighted prioritization of alternatives regarding the control criteria.

2.2.6. Optimization

The process of optimizing systematically and simultaneously a collection of objective functions is called *multi-objective optimization (MOO)* or *vector optimization* (Marler and Arora, 2004 [251]). MOO is used whenever certain solutions or parts of solutions exist, the values of the solutions with respect to objectives are known, there are some constraints for selecting solutions, but the complexity of the optimization problem hinders a human to figure out the optimum, or an automated selection is desired. The optimization problem can be complex due to the sheer number of solutions, the number of constraints, and / or the number of relations between solution

parts. The following definitions are used in the rest of the chapter:

$\mathbf{F}$ Vector of objective functions
(point in the criterion (problem) space) (2.1)

$F_i \in \mathbf{F}$ The i-th objective function (2.2)

$\mathbf{F}^o$ Vector of utopia points
(optimizing the collection of objective functions) (2.3)

$F_i^o \in \mathbf{F^o}$ The utopia point for the i-th objective function (2.4)

$\mathbf{G}$ Vector of inequality constraints (2.5)

$g_j \in \mathbf{G}$ The j-th inequality constraint (2.6)

$\mathbf{H}$ Vector of equality constraints (2.7)

$h_k \in \mathbf{H}$ The k-th equality constraint (2.8)

$\mathbf{x}$ Vector of design (decision) variables
(points in design (solution) space) (2.9)

$\mathbf{w}$ Vector of weighting coefficients / exponents (2.10)

$w_l \in \mathbf{w}$ The l-th weighting coefficient / exponent (2.11)

The general multi-objective optimization problem is posed as follows (note that $\geq$ constraints can be easily transformed to $\leq$ constraints. The same is true for maximization objectives.) :

$$Minimize\ \mathbf{F}(\mathbf{x}) = [F_1(\mathbf{x}), F_2(\mathbf{x}), \ldots, F_m(\mathbf{x})]^T \tag{2.12}$$

$$subject\ to\ g_j \leq 0 \qquad j = 1, 2, \ldots, n \tag{2.13}$$

$$subject\ to\ h_k = 0 \qquad k = 1, 2, \ldots, o \tag{2.14}$$

where m is the number of objective functions, n is the number of inequality constraints, and o is the number of equality constraints. $x \in E^q$ is a vector of design variables (also called decision variables), where q is the number of independent variables x_i with type E, which can be freely chosen. $F(x) \in E^k$ is a vector of objective functions $F_i(x) : E^q \rightarrow E^1$. $F_i(x)$ are also called objectives, criteria, payoff functions, cost functions, or value functions. The feasible design space $\mathbf{X}$ (often called the feasible decision space or constraint set) is defined as the set $\{x \mid g_j(x) \leq 0, j = 1, 2, ..., n \ \wedge \ h_i(x) = 0, i = 1, 2, ..., o\}$. x_i^* is the point that minimizes the objective function $F_i(x)$. An F_i^o utopia-point is the value attainable at best for $F_i(\mathbf{x})$ respecting the constraints. The total optimum is the value attainable at best for $F_i(\mathbf{x})$ not respecting the constraints.

Definition Pareto Optimal: A point, $x^* \in X$, is Pareto optimal iff there does not exist another point, $x \in X$, such that $F(x) \leq F(x^*)$, and $F_i(x) < F_i(x^*)$ for at least one function from F.

Chapter 3

Background on Scientific Methodology

In this chapter we will discuss how research in information systems and software engineering should be conducted (Section 3.1), and which strategies, processes and methods are available to conduct the research (Section 3.2[1]). This way the reader will get an idea what constitutes research in information systems and software engineering and how this work was influenced by the best practices available for this task. We conclude this chapter in Section 3.3[2]. In the next chapter (Chapter 4[3]) we will go into detail which methods have been chosen for the research presented in this work.

3.1. Research in Information Systems and Software Engineering as Design Science

Research in information systems and software engineering, and therefore also in requirements engineering, is commonly recognized as conducting design science (Wieringa, 2014 [388]; Peffers et al., 2007 [308]; Hevner et al., 2004 [184]). Design science in this context is defined by Wieringa (2014 [388]) as

> Design science is the design and investigations of artifacts in context. The artifacts we study are designed to interact with a problem context in order to improve something in that context.

Note that the term artifact is interpreted broadly by Wieringa (2014 [388]). An artifact is not limited to a physical entity, but everything which is suitable to improve a certain situation is considered to be an artifact including a piece of software, a method, an algorithm, a process, and so forth. The term context includes everything the artifact directly interacts with to improve a problem-bearing situation. A context, for example, can include people, hardware, existing processes, norms, organizations, and so forth (Wieringa, 2014 [388]). Hence, in design science we are not only interested in extending the knowledge about phenomena in a field, which is the case for basic sciences such as, for example, physics, nor are we only interested in solving a problem once, which is the case in applied research in the industry. In design science we strive for solving a general problem in its context with a constructed solution. Likewise we aim at extending the body of knowledge about the problem and solution space and its context in general.

[1] Page 50
[2] Page 66
[3] Page 67

3.1.1. Characteristics of Design Science

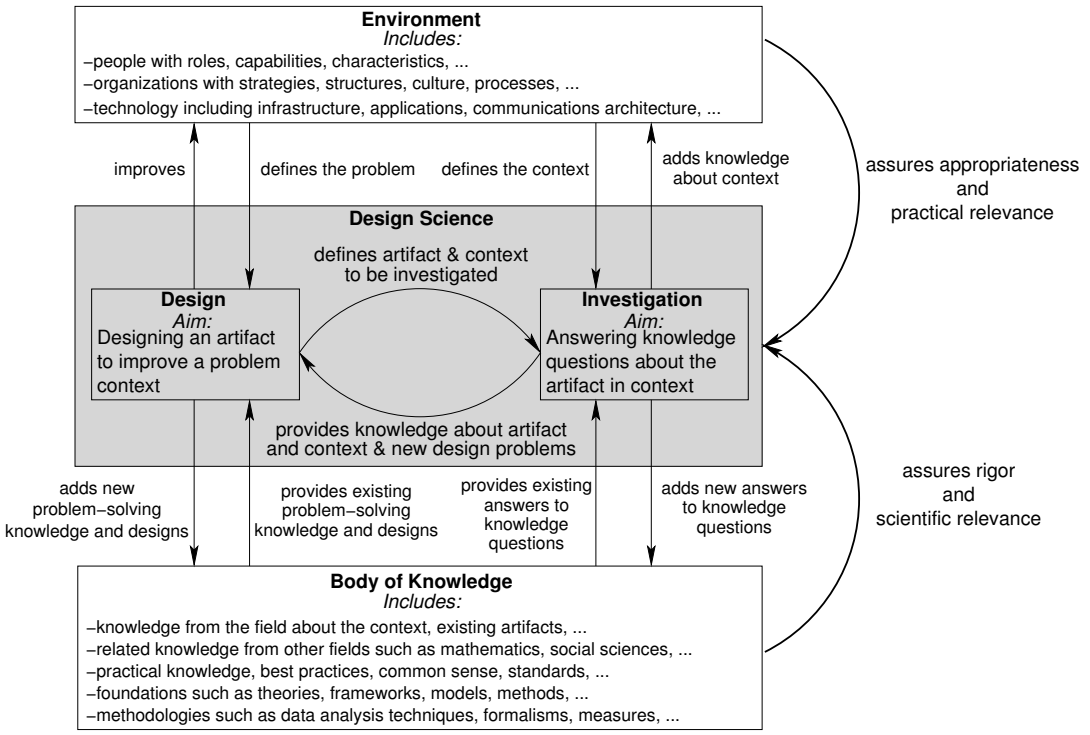


Figure 3.1.: *Design Science Framework (based on Wieringa (2014 [388]), and Hevner et al. (2004 [184]))*

Figure 3.1 shows a framework which helps to understand the characteristics of design science. Conducting *design science* includes two general blocks of activities. First, the *design* of the actual artifact, which aims at improving or solving a certain problem. Second, the *investigation* block, which aims at answering certain questions about the artifacts in their context. The design activities *define the artifact and the according context*, which is then the topic of the investigations. The result of these investigations *provide knowledge about the artifact and context, and, maybe, new design problems*. Hence, we have a feedback loop which is iterated until we found a sufficient solution.

Design science as such does not exist in isolation, but is embedded in an *environment* and the existing *body of knowledge*. The environment can include *people with different characteristics, capabilities* and so forth, *organizations which differ in their structure, culture,* and so forth, and *technological elements including the infrastructure, existing applications* and so forth. Within this environment the problem our artifact shall solve becomes manifest. Hence, the environment *defines the problem* we try to solve within our design activities. Ultimately, the designed artifact is supposed to *improve* the situation in the environment by solving the problem. In consequence, when we investigate the artifact in its context, the *context is defined* by the original environment the problem stems from. While investigating the artifact in its context, we might also learn new knowledge which is relevant for the environment and which we can feed back to the environment. Hence, the investigation of the artifact also *adds knowledge about the context*. By taking into account the environment, we *assure the appropriateness and practical relevance of our artifact*.

But only considering the environment might not generate any valuable results from a scientific point of view, even when we succeed to solve the problem which exists in practice. Hence, as counterpart to the environment, we have to consider the scientific body of knowledge. This body of knowledge provides us with already existing answers and knowledge about the context

and existing artifacts, related knowledge from other fields, practical knowledge, foundations, and methodologies. The *existing problem-solving knowledge and designs provided* by the body of knowledge helps us when designing our solution. For example, we can avoid known pitfalls, solutions that are known to be not sufficient, solutions we can build on and so forth. Of course, while designing our artifact we *add new problem-solving knowledge and designs to the body of knowledge*. Also for the investigation of the artifact the body of knowledge is indispensable, as it provides existing answers to knowledge questions, such as "How to analyze the data?", "How to conduct an experiment", but also "Do we already know how a specific entity influences our artifact?". Additionally, by sticking to scientific standards and building on existing knowledge, we make our results usable for other scientist. Hence, by considering the body of knowledge we *assure the rigor and scientific relevance* of our research.

3.1.2. Research Goals in Design Science

Every design science research project starts with one or more research question(s) and the goal to answer these question(s). According to Wieringa (2014 [388]) there are different types of research goals. First of all, there are goals which express the need to solve a *design problem*. Such general goals are mostly focusing the *artifact and its (re)design* to achieve some new / additional properties. A second kind of design problem and related goal arises often within the research process itself. It is about an *instrument design* which shall be used to investigate the artifact in its context. An instrument design problem always arises whenever no standardized or best practice methods and related measures for assessing an artifact are available, which nowadays is often the case in software engineering research. Another general research goal which arises often while conducting design science is related to *prediction problems*. The results from an investigation often need to be interpolated to a more general setting or to the future. Such a prediction can be very challenging and the method to answer it might be a scientific contribution itself. A third type of general research goals is about answering *knowledge questions*. Answering such questions can be divided into *analytical problems*, which can be solved using mathematics, logic, and so forth, and *empirical problems* for which we need to analyze data from the real world to answer them. Again, we can distinguish two kinds of empirical problems. First, problems for which the answer is *descriptive*. Hence, we only have to describe how the world looks like regarding the question. For the second kind, the answer has to be *explanatory*. In this case, we not only have to describe what happens but also provide a reasoning why it happens. Orthogonal to these types of research question is another characteristic. A question might be open or closed. For an *open question*, we do not have any idea what the answer might be, or are not able to enumerate all possibilities. For a *closed question*, we know all the possible answers, but do not know which one is correct. All these different kinds of research goals and according questions can be part of a research project, all of them are of equal importance, and we should be aware of them and make them explicit whenever a new goal arises.

What we see from this short discussion on research goals (for an elaborated discussion see Wieringa (2014 [388])), that even when we start from one research goal, which is in most cases a design goal, this initial goal might lead to other goals or be decomposed while conducting the research. Hence, deriving and answering all question related to those goals is an iterative process.

3.1.3. The Design and Empirical Cycle

But even when researchers know the blocks and activities of design science and the relations between these blocks, which are important for successful design science, and they are aware of their research questions and their types, conducting a design science project is not trivial.

Some authors, for example Wieringa (2014 [388]), and Hevner et al. (2004 [184]), try to support researchers in design science by providing further means to keep their research on track.

Hevner et al. (2004 [184]) provide seven guidelines researchers should follow and keep in mind (For an elaborated description of the guidelines see Hevner et al. (2004 [184])):

Design as an artifact Whenever we are doing design science, we are going for at least one artifact. Every activity taken by the researchers has to be related to this artifact. Hence, while conducting the research, the researchers should check each activity if it improves the artifact. If an activity has no impact on the artifact, it is out of scope.

Problem relevance The motivation for design science should not be pure curiosity, but there should be an existing problem which shall be solved. At best, this problem should be found in the "wild" and should not only be a scientific problem. If there are many problems, researchers should go for the most important and pressing ones for which hardly any or no treatment exists.

Design evaluation The efficacy of an artifact regarding a certain problem has to be shown. First on a theoretical level, but later also in practice. A final treatment should never be proposed without any kind of evaluation.

Research contributions You are not doing design *science* if your work does not contribute to the areas of the design artifact itself, design foundations, or design methodologies. If you just got a treatment, but no knowledge to share which improves the body of knowledge, you neglected the investigation part of design science.

Research rigor Design-science research relies upon the application of rigorous scientific methods, especially for evaluating the designed artifact. The application of rigorous methods assures the significance and confirmability of results.

Design as a search process Design science is not a top down or waterfall process. Improving the artifact is an iterative process in which the solution space is explored and searched.

Communication of research The result of design science has to be presented to different audiences (for example, other researchers, the management which has to approve a treatment, and the technicians which have to apply the treatment). This has to be kept in mind when describing and presenting the final treatment.

Wieringa (2014 [388]) provides a more elaborated guidance by proposing two kinds of intertwined cycles one can follow when conducting design science (see Figure 3.2, which mainly is an aggregation of the cycles described in Wieringa (2014 [388]), but also includes aspects of Wieringa and Moralı (2012 [389]), Hevner et al. (2004 [184]), Runeson et al. (2012 [326]), Wohlin et al. (2014 [393]), and Peffers et al. (2007 [308])). First of all, there is the *design cycle* in which we create the artifacts for solving our problem at hand. As a problem might be too complex to be handled by only one artifact, we might create several artifacts, each solving some sub-problem, and combine them. The combination of artifacts and the way of using them in a specific context is regarded as treatment by Wieringa (2014 [388]). In other works the term solution is used as a synonym for treatment. Note that for Wieringa (2014 [388]) there is a difference between those two terms. From his point of view, a treatment implies the possibility of creating new problems while the term solution does not. From our point of view, a solution can also bear new problems. Hence, we do not distinguish these two terms. The second cycle is the empirical cycle. In this cycle, the designed artifacts are investigated. Hence, we try to answer all important knowledge questions regarding our problem and the related possible artifacts and treatments. This in particular includes the scientific investigation of the artifacts themselves. A

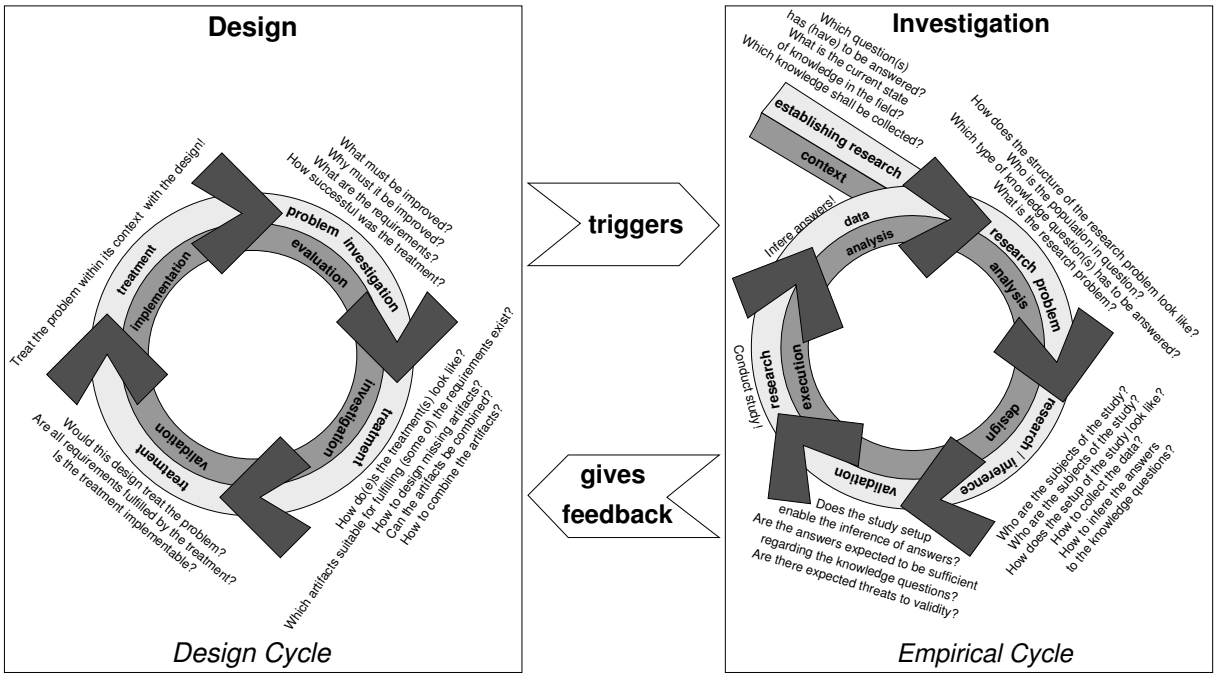


Figure 3.2.: *Design Science Framework (based on Wieringa (2014 [388]), Wieringa and Moralı (2012 [389]), Hevner et al. (2004 [184]), Runeson et al. (2012 [326]), Wohlin et al. (2014 [393]), and Peffers et al. (2007 [308]))*

meaningful combination of these two cycles enables us to successfully conduct any design science project.

3.1.3.1. Design Cycle

The design cycle is structured into the activities *problem investigation / evaluation*, *treatment investigation*, *treatment validation*, and *treatment implementation* Wieringa (2014 [388]). The activities and design cycle themselves are not that different to design cycles of other engineering fields such as[4] mechanical industry products (Cross, 2008 [107]), and industrial systems (Hall, 1962 [168]). Of course, the activities are adapted and specific to the software engineering field but their high-level purpose is the same as in any other field where design problems have to be solved. Hence, this cycle shows the best practice for solving design problems.

3.1.3.1.1. Problem Investigation / Evaluation The activity *problem investigation / evaluation* is about understanding the problem and its relation to the treatment. Hence, in the first iteration we focus on the leading questions:

What must be improved? Here, we try to understand the problem which occurs in a particular context. This includes the investigation of the already used treatments, their relation to the context, and in which situations the outcome is insufficient.

Why must it be improved? Answering this question ensures the relevance of the problems. In detail, we have to investigate how often the problem occurs, which negative impact it has, and which relevance it has in its context.

[4]Wieringa (2014 [388]) enumerates some more examples from many different fields.

What are the requirements? In case we are sure that we have identified a relevant problem, we have to elicit the requirements a treatment has to fulfill to improve the situation caused by the problem. These requirements are of particular importance, as they help us to scope our research and focus our design on the important parts for solving the problem.

In the following iterations, we might reconsider the first three questions as we might have missed, for example, a requirement, but a fourth question becomes important:

How successful was the treatment? Even though we try to validate the chosen treatment before implementing it, it might turn out that the outcome of implementing the treatment is different from what we expected. Hence, in each further iteration we have to reflect on the actual performance of our treatment and try to identify weaknesses and further problems.

3.1.3.1.2. Treatment Investigation After we understood the problem and the context in which it occurs, we start the *treatment investigation*. This in particular includes the design of new artifacts and their combination to get a new designed treatment. Hence, the following questions are of central importance:

Which existing artifacts are suitable for fulfilling (some of) the requirements? To keep the effort low and to base our treatment on well established and investigated treatments, the first task for us is to search and find already existing artifacts which might solve our problem or parts of it. As already existing artifacts might not fully fulfill our requirements, we might have to apply some adaptation to our problem and context.

How to design missing artifacts? Especially in a research setting we often face situations in which we cannot find any existing artifact for some of our requirements which have to be fulfilled for solving the problem. Hence, we have to decide how to design completely new artifacts.

Can the artifacts be combined? In some cases artifacts exclude each other or do not work well in combination. Hence, we have to investigate the possible combinations.

How to combine the artifacts? From the possible combinations we have to chose the most sufficient ones to cover all of our requirements. This often includes the design of an integrative artifact, which guides the combination, for example, by defining an overall process and the interfaces between artifacts.

How do(es) the treatment(s) look like? The integrative artifacts already represent some kind of treatment. The only thing missing is the adoption to our specific context at hand and the shape of the interaction between context and treatment.

3.1.3.1.3. Treatment Validation In the words of Wieringa (2014 [388])

> To *validate a treatment* is to justify that it would contribute to stakeholders goals if implemented.

Hence, validation in this sense is not used as strict as, for example, verification in mathematics or formal methods. Validation in the terms of Wieringa (2014 [388]) includes every activity which allows to reason about the fulfillment of our requirements and the respective goals. This can be logical reasoning, discussions with experts, but also any experiment which provides an indication on the sufficiency of the treatment. The aim here is to assess the treatment before actually take the effort to implement it. Again, there are some guiding questions we can follow:

Would this design treat the problem? To make sure that the treatment actually addresses the initial problem, we have to search, collect, and summarize all the reasons, indicators and so forth which imply the sufficiency of the selected treatment.

Are all requirements fulfilled by the treatment? Part of assessing the treatment for its sufficiency is to recheck it against the requirements elicited in the first activity.

Is the treatment implementable? Even though the requirements are fulfilled and the treatment is sufficient to handle the initial problem, it might happen that a treatment is still not implementable. There might be several reasons for this, for example, the effort of using the treatment might be to high, the costs might exceed the budget, or the context in which the treatment shall be implemented cannot be influenced in the necessary way.

3.1.3.1.4. Treatment Implementation The *implementation of a treatment* is the application of the treatment to the original problem context (Wieringa, 2014 [388]). But this activity is not only about the implementation but also about adjusting the treatment, observing how the treatment is accepted and adapted by the context, and collecting occurring new problems caused by the treatment or unsolved parts of the original problem.

3.1.3.2. Empirical Cycle

The empirical cycle is structured into the activities *establishing research context*, *research problem analysis*, *research / inference design*, *validation*, *research execution*, and *data analysis* (see Figure 3.2). This cycle is also proposed by Wieringa (2014 [388]) and is in line with the best practices and state of the art in the field of empirical research in software engineering as, for example, described by Wohlin et al. (2014 [393]), Runeson et al. (2012 [326]), and Peffers et al. (2007 [308]). Note that Wieringa (2014 [388]) calls this cycle empirical cycle, as most of the time we apply empirical methods throughout the cycle. However, this cycle is not limited to empirical research only, but every kind of investigation method. This also includes, for example, logical deduction methods, mathematical analysis and so forth. Hence, the empirical cycle helps us to structure any kind of investigation and gives us guidance based on the best practices in the field.

3.1.3.2.1. Establishing Research Context Before starting the actual investigations and therefore the empirical cycle, the *context of the research* has to be *established*. The topic of an investigation might be about the treatment itself but also about any knowledge question which is relevant for understanding the problem, understanding the context of the problem, or about existing treatments. For establishing the context some questions should be considered and answered:

Which question(s) has (have) to be answered? Before starting any research activities and spending any effort, it should be clear which knowledge question(s) shall be answered for which reason(s). These questions scope the following activities and are of particular importance for the selection of sufficient methods for answering them. Without clear questions, the result of an empirical cycle might only give sufficient feedback to the design cycle by chance.

What is the current state of knowledge in the field? Before we start searching for (an) answer(s) by ourselves, the existing body of knowledge should be searched for existing answers. This ensures that we build upon the effort others spent instead of spending effort on an already known answer. Even though we might not find the direct answer(s) to our questions, we might get aware of knowledge which can guide our own investigations.

Which knowledge shall be collected? It is not only important to have the questions, but also to define which knowledge we would need to give an answer. The empirical cycle is then executed to collect this knowledge.

3.1.3.2.2. Research Problem Analysis When doing the *research problem analysis* we try to understand which research problem has to be solved to collect the knowledge which is needed to answer our research questions. In case we are not in the first cycle but any further cycle, we also have to determine which parts of the research problem is already solved by the preceding iterations and which parts are unsolved and why they remained unsolved.

What is the research problem? Hence, the overall question is what research problem we are facing. Do we only want to collect existing data, do we only want to observe something, do we want to test something, and so forth? Answering this question is clearly bound to the knowledge needed to answer our research questions.

Which type of knowledge question(s) has to be answered? Even though we already have formulated our research questions, we might not have determined their type. In Section 3.1.2 we have seen that there are different types of knowledge problems and according questions. Of course, the different types also imply different research problems. The research to be done for an analytical research problem is different to what we have to conduct for answering an explanatory question.

Who is the population in question? This is a question which is often neglected but nevertheless very important. Who are the subjects who are sufficient for answering our questions? Do we need experts in a certain field or are students sufficient? To be able to determine a sufficient population of a research problem we need knowledge about and the explicit definition of important characteristics of the population which is in question in the original research question. It might be also of relevance what we know about the distribution of certain characteristics within our population.

How does the structure of the research problem look like? In this question, we try to determine which effects and part of the context might influence the result of our research. Hence, we are looking for all the important variables and their relations to each other. Variables are not only bound to the population, but every entity that exists in the context.

3.1.3.2.3. Research / Inference Design Next, we have to *design* our *research* and the way we plan to *infer* results from the research. This includes the selection of appropriate research and analysis methods. In this activity we shape the study we want to conduct to gather the knowledge needed to answer our research questions.

Who are the subjects of the study? From the previous steps we know the population in question and the important characteristics of this population. But we might not be able to conduct a study with subjects which completely cover the original population in question. Hence, we have to explicitly state how we select the subjects, how they relate to the original population, and which impact differences between the subject population and the original population has on the results and their interpretation.

What are the objects of the study? Here, we define the treatments and cases used to investigate the research problem. Also the context of the study has to be defined. Again, it is important to distinguish between the context defined by the original problem, and the context we establish for conducting the study. Like for the population of the research, the context chosen for the research might only partially cover the original context and we should be aware how this impacts, for example, the possibilities to generalize results.

How does the setup of the study look like? To answer this question, we have to make the overall setup of the study explicit. We have to describe the methods we chose to conduct the study, the methods to be used to obtain data, and methods to be used to derive knowledge from the data.

How to infer the answers to the knowledge questions? We also have to state how to infer the answers from the derived knowledge, to which extent these answers cover the original questions, and which limitations we expect regarding the inference.

3.1.3.2.4. Validation Before actually conducting the study, we have to assure that the effort to be taken is expected to pay off. Hence, we have to *validate* our study. For validating a study design, only some basic methods or guidelines are available. The best practice for this activity is to reason about possible threats to validity, and to involve other researchers to assess a study. Some questions might help to conduct the validation:

Does the study setup enable the inference of answers? Here, we have to check whether the study design matches the needs for the chosen inference method. For example, some statistical methods only allow one variable to vary. If within the study setup more than one variable is expected to vary, the study design does not match the requirements of the statistical method.

Are the answers expected to be sufficient regarding the knowledge questions? The study design might restrict the investigated subjects, objects, and context. This and the threats to validity also restrict the expressiveness of the results and the related answers. We have to make sure that still the answers are expected to be sufficient regarding the knowledge questions.

Are there threats to validity? One important question when assessing a study design is the question of threats to validity. Which things might have influenced already the study design in an inappropriate way, or might influence the execution of the study and therefore jeopardize our results? And in case there are threats to validity, are we able to mitigate those threats? In case any threat is left over, we have to assure that this threat cannot invalidate the overall results. If such a threat exists, we are better off with not conducting the study or choosing a different design.

3.1.3.2.5. Research execution and Data Analysis In case we are confident that a study design is sufficient to collect the required answers, we proceed with *executing the study*, and analyzing the resulting data. These activities also include to monitor the threats to validity which might appear during the study execution or data analysis. Here, the question is if the threats really show up, to which extent they manifest, and which impact they have on the results. We also have to protocol the execution of the study, to be able to judge later which problems occurred and to reason why certain expected results did not show up.

3.1.4. Design Science does not Follow a Strict Process

The two cycles seem to enable one to form a strict process for a design science problem which includes some repetitions of activities. This is not the case, which is stressed by many authors (Wieringa and Moralı, 2012 [389]; Wieringa, 2014 [388]; Hevner et al., 2004 [184]; Wohlin et al., 2014 [393]; Runeson et al., 2012 [326]). The opposite is the case. While we have two peaks of knowledge regarding a specific design problem (see Figure 3.3), the design knowledge and the investigation knowledge, this knowledge increases while conducting design science, but the use of the design and empirical cycle to increase this knowledge is far away from being strict. Some

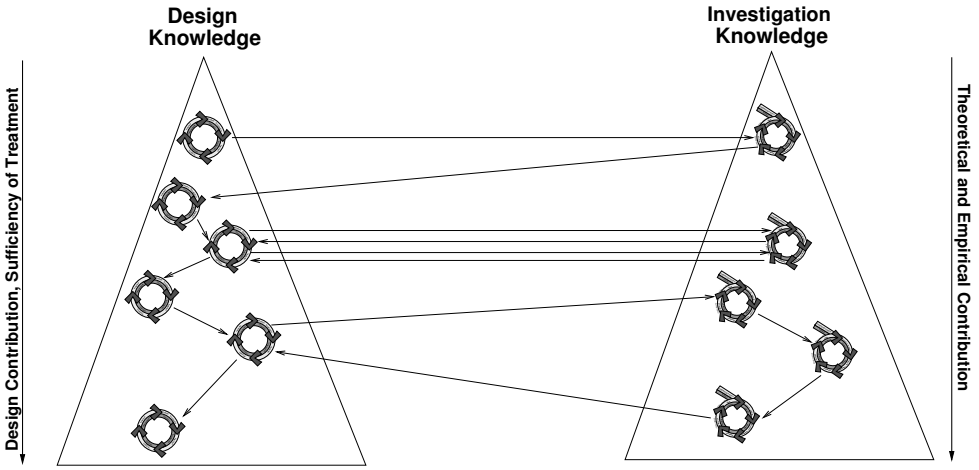


Figure 3.3.: *The Twin Peaks of Design Science (adapted from Nuseibeh (2001 [289]))*

iterations within a cycle might be shortened by leaving out activities. For example, if we are already investigating the treatment and we notice that parts of the problem are still too vague for selecting / designing a treatment, we will skip the treatment validation and implementation, and start a second iteration by conducting a more detailed problem investigation. Also the intertwining of design and empirical cycles can come in different shapes. One design cycle might trigger an empirical cycle and the feedback of this empirical cycle lead to a new design cycle. This might happen if the results of an experiment indicate that we did not understand the problem correctly in the first place. One design and one empirical cycle might be closely intertwined leading to a situation in which you iterate them simultaneously. For example, establishing the research context helps you to understand the problem and so forth. We might also iterate some design cycles without having an empirical cycle in between. Or we have some iterations of the empirical cycle until we get the feedback we need to resume the design cycle.

3.2. Research Processes and Methods

While the two cycles and their combination structure the overall research, they do not define any strategy, process, or method to be used for a certain activity. Here, we have to select the appropriate strategy, process, and method(s) for each activity considering the particularities of our research problem. Some authors distinguish between methods which are suitable for the design cycle (Section 3.2.1) and methods which are suitable for the empirical cycle (Section 3.2.2) (Braun, Wortmann, Hafner, and Winter, 2005 [63]).

But for some processes and methods this separation is not clear cut (Wieringa, 2014 [388]; Hevner et al., 2004 [184]). For example, a survey might be used for problem investigation in a design cycle as well as it might be used for research problem analysis in an empirical cycle. Moreover, a survey might also be used for treatment validation. From this example we see that some methods can belong to different activities and cycles. In the following, we will give a short overview of processes and methods which are currently regarded as best practices in design research. We will classify them as design or empirical processes and methods based on their main purpose, but the reader should keep in mind that some of the processes and methods can be used for different purposes.

3.2.1. Design Processes and Methods

Designing artifacts and treatments is a process involving creativity (Braun et al., 2005 [63]; Bayazit, 2004 [34]; Shneiderman, 2007 [347]; Nguyen and Shanks, 2009 [282]). The research on design and creativity has already a long tradition and there are many works proposing processes to deal with this topic (for example, the works of Wallas (1926 [384]), Osborn (1963 [301]), and Shneiderman (2000 [346])). As these works are domain agnostic, they can be applied and used for any design problem including design problems in software engineering (Nguyen and Shanks, 2009 [282]).

There are three different views on creativity which differ in their focus and therefore used methods. There is the *inspirational view*, the *structural view*, and the *situational view* (Shneiderman, 2007 [347]; Nguyen and Shanks, 2009 [282]). When taking the inspirational view, breaking away from familiar structures is enabling creativity. Hence, the aim is to create situations in which one is able to investigate a problem from a completely different and unfamiliar point of view. The dominant process for taking the inspiration view is the process of Wallas (1926 [384]). It consists of the steps *preparation*, *incubation*, *illumination*, and the *verification and expression of insights*. In the preparation phase, one has to decide which methods to use, at which place(s) the research shall happen, and which means shall be used. Common methods are brain writing / storming, mind mapping, bionics, analogy technique, provocation technique, thinking out aloud, and so forth. As all senses should be triggered, it is also important to change to an uncommon place, or even change the place several times (Wallas, 1926 [384]). Many methods make use of certain means which enable one to take a different view and rethink the problem in a new way. For example, picture books can be a means to provoke creativity, but also a certain pet, a bio-system, or a story. In the incubation step, we apply the method at the specified place(s) using the means. We play around and are not restricted in any way. We are not searching for a solution, but for relations, new combinations, and new views on our problem. The incubation step is followed by the illumination step. This is a completely unstructured step, which cannot be planned or forced to happen at a certain time. The solution suddenly appears triggered by the new impressions gained in the incubation step (Wallas, 1926 [384]; Nguyen and Shanks, 2009 [282]). But it might even happen that the new impressions do not trigger an illumination at all. But in case we have an illumination, we have to verify its applicability and usefulness and express the new gained insights.

The structural view on creativity does not rely on a sudden illumination but on an planned and structured exploration of the solution field. From a structural point of view a solution to a specific problem has to cover different aspects and parts of the problem to be sufficient. For each of the aspects and parts, only a finite set of applicable options for covering them exists. Hence, finding a new solution is about finding the options, and combining them. A well established process for enabling creativity in a structured way is the Creative Problem Solving process (*CSP*) (Osborn, 1963 [301]). The process consists of six steps. The first step is the *mess-finding*. In this step, we define the high level objective(s) of our new design. For the structured search and the scoping of the solution space, it is very important to understand the problem and its important aspects and parts. Hence, instead of directly searching for solutions we have to analyze the problem. For this purpose, we are searching for data in the step *data-finding*. The data should describe the problem as well as the context in which it appears. In the next step *problem-finding*, the data is analyzed to pinpoint the specific problem, and the problem is described in a detailed way. After we have understood the problem, we can start to *search for ideas*. Hence, we decompose the problem to atomic sub-problems, and collect the according options to solve a sub-problem. Doing so, we span the solution space in which we will search for the best combination of options in the next step *solution-finding*. In this step we start to combine the different options to a complete solution. This way we might find several solutions.

Hence, we have to analyze them if they are acceptable for solving our problem. This is done in the last step, *acceptance-finding*. Well known methods which can help to implement this process are morphological boxes, relevance tree analysis and so forth. There are methods which already detail each step such as, for example, the TRIZ method (Webb, 2002 [385]).

The situational view is based on the hypothesis that creativity roots in social interaction. Different people bring in different views, knowledge of the solution space and so forth, and if the interaction between them is fostered in a sufficient way, they will be creative together. Hence, the creativity problem is about finding the right people and creating an environment which enables communication and knowledge transfer. This view is relatively new in the field of design and creativity. One of the first attempts to structure the process of situational creativity was formulated by Shneiderman (2000 [346]). But this process is basically a broadened CSP which emphasizes tools and methods which support communication and knowledge exchange, such as, for example, wikis, electronic mind map boards, forums, but also specific kinds of activities which are similar to the communication activities in an agile process such as SCRUM (Takeuchi and Nonaka, 1986 [366]).

Note that all of the processes mentioned in this section already cover a full design cycle. The names of activities and the granularity of activities differ, but in the end they all cover the same things to do in the same order.

Recently, some researchers argue that one should not take only one view, but combine the different views (Nguyen and Shanks, 2009 [282]). Hence, the use of creativity processes and methods is not mutually exclusive and should not be treated this way. Additionally, the success of a creative process cannot be predicted and is not guarantied at all. Hence, researcher should be creative in the way they use and combine creativity processes and methods.

3.2.2. Empirical Strategies, Processes, and Methods

According to McGrath (1984 [265]), and Stolen and Solheim (2007 [362]) there are several strategies to conduct empirical research. The different strategies also imply the use of different methods. Figure 3.4 shows the strategies. It also shows that there are three important dimensions which have to be considered when choosing a strategy (McGrath, 1984 [265]; Stolen and Solheim, 2007 [362]):

Precision How precisely can the behavior under investigation be measured?

Generality Is the resulting evidence generalizable over the target population?

Realism Is the situation and context in which the evidence is collected comparable to the situation and context under investigation?

McGrath (1984 [265]) states that in an ideal world, we would maximize all dimensions at once when conducting empirical research, but, unfortunately, maximizing all of them at a time is not possible (McGrath, 1984 [265]; Stolen and Solheim, 2007 [362]). The reason is that there are two factors which vary for each strategy and its characteristics regarding the dimensions. The first factor is to which extent our research actions influence the observed behavior and therefore our results. Here the scale reaches from very *obtrusive* strategies to *unobstrusive* strategies. While obtrusive research operations allow a high precision, they tend to impact generality and realism negatively. The second factor is the system behavior we are investigating. This scale reaches from a *universal* behavior to a very specific one for a *particular* case. While a particular case enables a high degree of realism, the precision and generality of the results is low. From this discussion we see that we have to be clear which dimension is the most important dimension for us, and select the strategy accordingly.

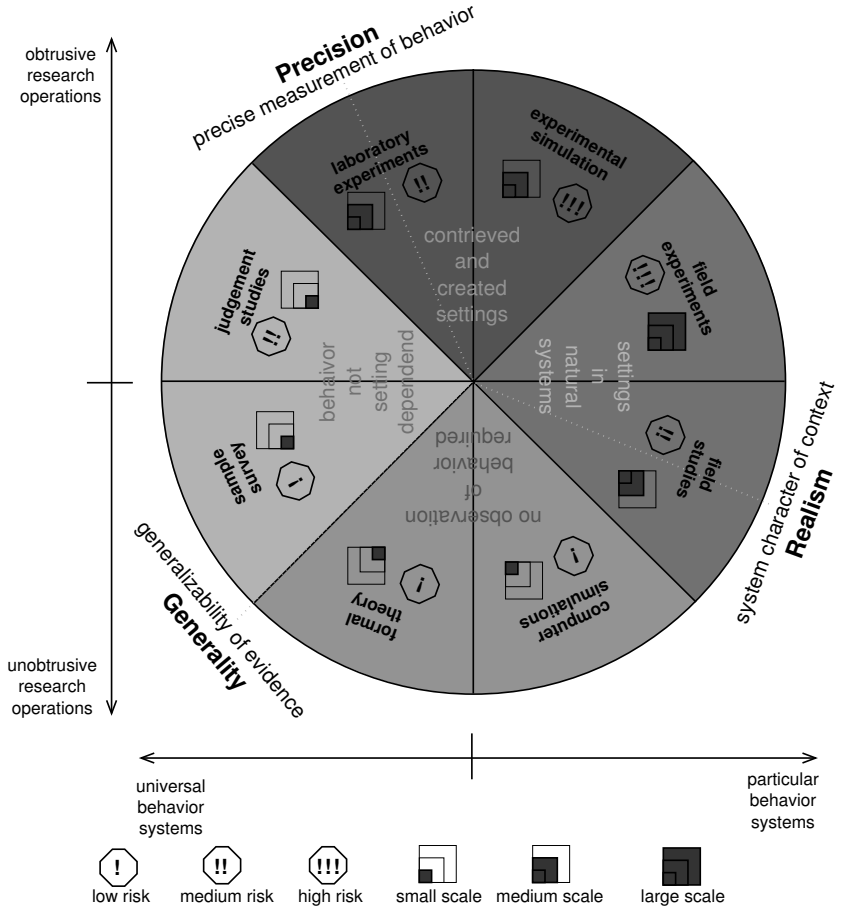


Figure 3.4.: *Empirical Research Strategies (based on (McGrath, 1984 [265]; Stolen and Solheim, 2007 [362]; Wohlin et al., 2014 [393]))*

According to Wohlin et al. (2014 [393]) different research strategies also differ in *risk* and *scale* (see Figure 3.4). The risk of applying a research strategy is that the research goals are not met, because, for example, the obtained data allows no conclusion, but a failure also might be that in case of a field experiment the complete process which shall be observed fails and no outcome is produced. A risk for a strategy also includes which measurements for certain threats exist and how easily they can be applied. The scale of a strategy expresses the time it takes to execute the strategy, the involved people, and the money and effort to be spent to realize the strategy. Note that the values given for risk and scale only represent the mean value for this strategy and represent the normal case (Wohlin et al., 2014 [393]). Hence, there is no strict order between the strategies regarding risk and scale, but the given values in Figure 3.4 can be used as orientation. As there are considerable differences in risk and scale between the strategies, Wohlin et al. (2014 [393]) recommend not only to consider these factors thoroughly, but also to start the research with strategies which have a low scale and risk and then move on to more sophisticated strategies. Because as we gain further knowledge, we are able to lower the risk for further research activities. This view is in line with other researchers (for example, Wieringa and Moralı (2012 [389]), Stolen and Solheim (2007 [362]), and Runeson et al. (2012 [326])).

According to McGrath (1984 [265]), and Stolen and Solheim (2007 [362]), the strategies are:

Laboratory Experiment In a laboratory experiment we try to replicate the real context and situation in an artificial and highly controlled environment. The controlled environment

allows us to select certain variables we would like to investigate while fixing all other variables. This results in a high precision, but even though we try to replicate the original context and situation, the fixing of variables leads to a low realism and as long as we do not vary the fixed variables, the generality of results is low, too.

Experimental Simulation An experimental simulation is like a laboratory experiment with the difference that no variable is fixed. Hence, the participants can act like in the real situation and context, but in the artificial environment. This enables a closer observation of the ongoing behavior, but lacks the ability to closely monitor one particular variable.

Field study In a field study, we try to make observations of a "natural" ongoing system, while not influencing it. The system should be the one we are interested in or comparable to the system we are interested in. This way, we get evidence with a high degree of realism, but as long as we only observe one system and this system is only a representative for a class of systems, the evidence is not easily generalizable. Additionally, as we avoid influencing the system, we cannot control any variable which leads to hardly any precision.

Field experiment A field experiment is a field study in which we decide to influence some variables, for example, the used artifacts. Hence, we gain some more control but sacrifice some realism.

Formal Theory When using a formal theory to evaluate some research, we formulate a theory which can be used to assess a certain artifact. Such a theory involves the modeling of the important parts for building the theory and their relation to each other. In most cases some mathematical notation is used for formulating the theory, but it is also possible to rely on pure argumentation. Afterward, some reasoning mechanism is used to test some artifact against the theory. As such a theory is an abstraction of the real world, it often lacks realism, and its precision can hardly be judged as long as it is not tested against reality or at least based on empirical data. Hence, a formal theory is not regarded as empirical strategy in most cases.

Computer Simulation A computer simulation is also a formal theory, but it can be executed in a certain way. Often the execution relies on some data from the real world. Hence, the realism is increased, sacrificing some generality.

Sample Survey In a sample survey, we try to collect as much data regarding our research problem as possible from already existing knowledge sources. A knowledge source can be a document as well as a person. The knowledge sources have to be carefully selected. It should be assured that their knowledge reflects the knowledge of the population for the problem at hand. Within a survey it is also of importance to not focus on a few knowledge sources, but a broad sample of sources. Hence, the precision of the results might be low as the data collected is second hand and we cannot, for example, influence the methods used for collecting the data. Also the realism regarding our research problem might be low, as situations and context in which the knowledge of the sources was collected can differ between the different sources as well as between each source and the context and situation we are interested in. But from a survey we get a high degree of generality.

Judgment Studies Judgment studies are also kind of a survey, but we focus on a minor number of knowledge sources, which directly fit to our research problem. A judgment study is limited to people as knowledge sources and is mostly conducted as an qualitative interview. For a judgment study we have more control over the sample and we try to select only those knowledge sources which are likely to fit our situation and context under consideration. Hence, we get a higher degree of precision, but sacrifice some generality.

	Laboratory Experiment	Experimental Simulation	Action Research (Special Kind of Field Experiment)	Field Experiment	Field Study	Survey
Primary Objective	Explanatory	Exploratory	Improving / Exploratory	Exploratory	Exploratory	Descriptive
Primary Data	Quantitative	Quantitative	Qualitative	Qualitative	Qualitative	Quantitative
Design Type	Fixed	Fixed	Flexible	Flexible	Flexible	Fixed
Contemporaneous Event	Yes	Yes	Yes	Yes	Yes	Yes
Type of Research Question	How, Why	How, Why	How, Why	How, Why	How, Why	Who, What, Where, How many, How much
Requires Control	Yes	Yes	No	No	No	No
Level of Control	High	Medium	Low	Low	Low	Medium
Difficulty of Control	Low	Low	High	High	High	Low
Level of Replication	High	Medium	Low	Low	Low	High
Cost of Replication	Low	Low	High	High	High	Low

Table 3.1.: *Further Characteristics of Research Strategies (based on (Runeson et al., 2012 [326]; Wohlin et al., 2014 [393]; Wieringa, 2014 [388]; Yin, 2009 [395]; Fenton and Pfleeger, 1998 [142]))*

Beside the information given in Figure 3.4, there are further means provided by researchers in the field of empirical research which help us to identify a sufficient research strategy. In Table 3.1 we summarized some characteristics of research strategies, which can help to select an appropriate strategy for the research problem at hand.

The first characteristic is the *primary goal* of a strategy. A strategy might aim at explaining certain effects. This includes the validation of certain hypotheses. In this case the strategy is *explanatory*. If it aims at discovering certain variables and relations it is *exploratory*. A strategy might also be used to *improve* a certain situation. Hence, it is not only used for gaining knowledge but has also a direct impact on a problem. Last, a strategy might be just *descriptive* and only describe the object under the investigation. The second characteristic is the type of *primary data* produced. This can be *quantitative* or *qualitative* data. The third characteristic is the *design type*. A research strategy might be *fixed*, which means that everything is planned before the execution of the research, and can and will not change during the execution. In contrast, some strategies are *flexible* allowing changes during the execution of the research. The fourth characteristic is about the studied phenomenon and if it might be a contemporaneous event or not. Hence, here the question is if the phenomenon can be isolated from its context it occurs in or not. Fifth, different strategies enable us to answer different *types of research questions*. Sixth, some strategies *require control* over the setting the phenomenon to be studied occurs in. Seventh, different strategies have different *levels of control* when executed, and, eighth, differ in the *difficulty of control*. The last two characteristics are about the possibility of replication (*level of replication*) and the *cost for a replication*. Before selecting a strategy, we should define the desired characteristics for our optimal strategy and then select the strategy accordingly.

For all of these strategies, different processes and methods are available. In this section we cannot discuss each and every process and method. Hence, we only discuss some important and commonly agreed research processes which cover different empirical research strategies. We will focus on strategies and methods which were of particular importance for deriving the results presented in this thesis[5]. In Section 3.2.3, the work of Wohlin et al. (2014 [393]) on the topic of experimentation in software engineering is introduced. In Section 3.2.4 we will discuss the matter of case studies in software engineering which includes the so called action research. This section is mainly based on the work of Runeson et al. (2012 [326]) about case studies and the work of Baskerville (1999 [32]) about action research. After looking at the case studies, we will have a look at theory building and explain the the grounded theory process by Glaser and Strauss (1967 [158]) (Section 3.2.5[6]). Last, we will discuss the structured literature review

[5]We will discuss the relation between the methods used and the results presented in this thesis in detail in Chapter 4.

[6]Page 62

process of Kitchenham (2004 [214]) as an example for a survey process.

3.2.3. Experimentation in Software Engineering

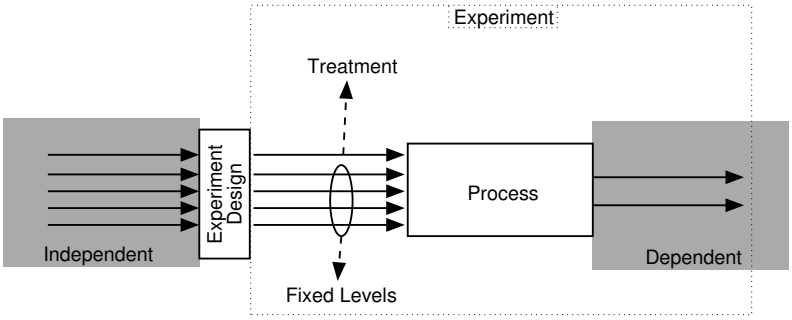


Figure 3.5.: *Explanatory Experiment (based on (Wohlin et al., 2014 [393]))*

According to different authors, for example, Wohlin et al. (2014 [393]), Runeson et al. (2012 [326]), and Wieringa (2014 [388]), there are two general types of experiments. The *explanatory or confirmative experiment* and the *exploratory experiment*. In an explanatory experiment, we strive to confirm a hypothesis we have about a certain variable, for example, the effect of an artifact on its environment. Figure 3.5 shows the generic setup of a such an experiment. First of all, there is the *process* we are interested in. A process might be any interaction and / or sequence of activities, which produces an output we want to understand and / or improve. The outcome is described by one or more *dependent variables* (Variables are depicted as arrows with solid lines). On the other end, we have *independent variables*, which influence the process and, in consequence, also the outcome. In an explanatory experiment, we are interested in one specific variable, which describes the effect of our *treatment*. Hence, we have to create an *experiment design*, which *fixes* all other variables to a specific *level* within the *experiment*. This way, all changes in the outcome of a process are due to changes in the variable under investigation. Hence, we can explain the influence of this variable which describes our treatment, which might validate our hypothesis or not.

In an exploratory experiment (see Figure 3.6), we are interested in understanding which variables have an impact on the outcome, what the effect on the outcome is, and how the variables are related to each other. Hence, we only fix those variables which are known to have an effect, but we are not interested in. Again, one of the variables which are not fixed might be related to a treatment we are interested in. But this is not required. It might also be the aim to understand the influences of an existing process before introducing changes to it. Hence, the

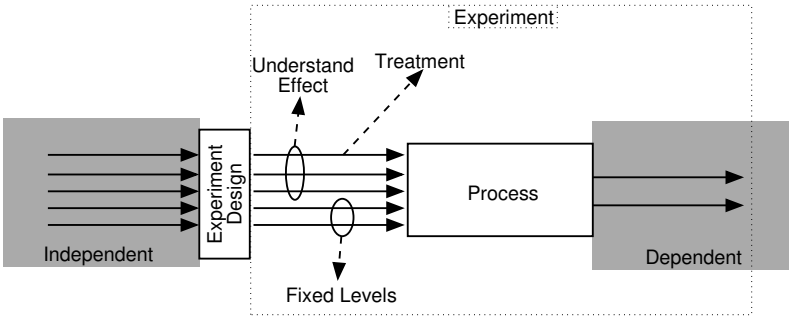


Figure 3.6.: *Exploratory Experiment (based on (Wohlin et al., 2014 [393]))*

main purpose of an exploratory experiment is to find out which variables have to be considered when conducting further research or when designing a new treatment.

For planning and conducting experiments of both kinds, one can rely on the guidelines, templates, and process as suggested by Wohlin et al. (2014 [393]). The general process described by Wohlin et al. (2014 [393]) consists of the steps *experiment scoping*, *experiment planning*, *experiment operation*, *analysis & interpretation*, and *presentation & package*.

Experiment Scoping When scoping the experiment, we have to clearly state the main hypothesis(es) under investigation and a first formulation of the experiment itself. For scoping the experiment Wohlin et al. (2014 [393]) suggest to use a template, which consists of the constituents Object of study (what is studied?), Purpose (what is the intention?), Quality focus (which effect(s) is (are) studied?), Perspective (from whose point of view is the experiment conducted?), and Context (where is study conducted?). The outcome of this step is a clear goal definition for the experiment.

Experiment Planning Wohlin et al. (2014 [393]) suggest to split up the experiment planning into the following steps:

Context Selection Selection of the environment in which the experiment will be executed.

Hypothesis formulation The hypotheses have to be formulated as formal statements, and it has to be stated which statements shall be tested based on which data derived from the experiment.

Variable selection In this step the variables under consideration have to be collected and defined. Leading question for this purpose are: Which (in)dependent variables have to be measured to test the hypothesis? How are they measured? Which scales does a variable have?

Selection of Subjects In this step, the process for selecting subjects for the experiment has to be designed. Important questions for this purpose are: Who are the participants? How to find them? Which characteristics does a participant have to have? How is the selection actually done?

Experiment Design For this step, we have to select the type of the experiment and design the overall process the experiment will follow. This step also includes the selection of the data analysis method. A data analysis method includes the preparation of data, and the descriptive statistical analysis as well as the hypothesis test which are necessary to interpret the results described the data.

Instrumentation For each step, we have to define or select the appropriate means to execute each step of the experiment design. The means selected can be ones used within the experiment itself or ones used for obtaining the data from the experiment.

Validity Evaluation Reasoning about the design and implementation, and if it is sufficient to reach the initial goal of the experiment. Different types of threats (See
Wohlin et al. (2014 [393]) for an enumeration of threats) to validity play a major role role within this activity.

Note that these steps do not form a strictly sequential process but iterations and jumping between steps are necessary till a good experiment design and solution settles.

Experiment Operation Of course, the experiment has also to be conducted. This includes the *preparation* of the experiment. Within the preparation the subjects are selected, and provided with all information they need know. The instruments chosen for the experiment activities also have to be prepared. The *execution* itself is not a problem when the experiment was designed

carefully. Nevertheless, we have to assure that we stick to the design. After the experiment, the data has to be *validated* to assure it reflects the experiment and free of flaws. The result of the experiment operation is the data obtained.

Analysis and Interpretation The first step to understand the results the obtained data might contain is to analyze the data using *descriptive statistics*. Such an analysis only describes and characterizes the data set, but does not allow any conclusions or hypothesis validations beyond informal reasoning. But it provides information about data points which may have to be removed, or gives first hints that some variables are redundant. Hence, the *data* set can be *reduced*. Afterward, we can conduct the actual *hypothesis tests*. The tests provide the results we are looking for and which have to be interpreted. Hence, the output of the analysis and interpretation activities are the actual conclusions which follow from the experiment.

Presentation and Package This step includes the presentation of the results as well as the experiment design itself. While the former is what is always done, as we are interested in the results and want to share them, the latter is often neglected. But the results can be only judged by others when they know the experiment design. Additionally, experiments often need to be replicated in order to make some results statistically significant.

Of course, for each of the steps, we have to choose further methods such as, for example, hypothesis test or data analysis methods. For example, Wohlin et al. (2014 [393]), Runeson et al. (2012 [326]), and Wieringa and Moralı (2012 [389]) and others enumerate some of these methods and describe them detail. The interested reader is referred to these sources, as a detailed discussion of all of these methods is not feasible within this thesis.

3.2.4. Case Studies in Software Engineering

The term case study is often stressed in recent publications in software engineering (Runeson et al., 2012 [326]). The term seems to be used whenever a specific case is used to investigate a certain artifact. Hence, the term case study is used for research in which an artifact is applied to a small toy example within an artificial environment, but it is also used for a real life case investigated in the context of a company (Runeson et al., 2012 [326]). Runeson et al. (2012 [326]) tries to straighten the use of the term case study. They survey some well known definitions for case studies from the social sciences, because the idea of case studies originates in this field. Runeson et al. (2012 [326]) analyze these definitions and aggregate them to definition which is of use in the field of software engineering:

> A case study in software engineering is an empirical inquiry that draws on multiple sources of evidence to investigate one instance (or a small number of instances) of a contemporary software engineering phenomenon within its real-life context, especially when the boundary between phenomenon and context cannot be clearly specified.

According to this definition, most studies which are presented as case studies in the software engineering literature are not case studies in the original sense. Using a case from the "real world" within in an artificial environment is not a case study, but an experiment (Runeson et al., 2012 [326]), because for a case study the investigation has to be executed in the "real-life context". Furthermore, a pure case study has to include several cases not only one ("multiple-sources of evidence"). But "single case"-case studies are widely accepted in the research community even though the conclusions drawn from such a study have to be treated with care (Runeson et al., 2012 [326]).

The definition of Runeson et al. (2012 [326]) covers the two empirical research strategies *field experiment* and *field study* (see Figure 3.4), as it neither forbids nor requires to influence the phenomenon to be observed. A very specific case of field experiment is the so called '*action*

research', which is explicitly considered as case study by Runeson et al. (2012 [326]). In the following we will use the term case study for a study in which we just observe the phenomenon in its original context (field study) or in which we changed the process to be observed by, for example, introducing a new treatment, but do not participate in the process itself (field experiment). If we take part we call this explicitly action research.

3.2.4.1. A Process for Case Studies

Runeson et al. (2012 [326]) propose a general process for conducting a case study which consists of the steps *case study design*, *preparation of data collection*, *collection of evidence*, *analysis of collected data*, and *reporting*. The process itself looks more or less like processes for other empirical strategies such as, for example, for experiments by Wohlin et al. (2014 [393]) or literature reviews by Kitchenham (2004 [214]). Even the general purpose of the steps and the activities within the steps are rather similar. But Runeson et al. (2012 [326]) stress the fact that within a case study it is possible to switch between the steps even when the study is already ongoing as a case study is meant to be flexible. The process of Runeson et al. (2012 [326]) is explained and summarized in the following.

Case Study Design In this step, we have to design the study with all necessary means. The activities do not differ much from the activities taken within *experiment scoping and planning* in the process of Wohlin et al. (2014 [393]) (see Section 3.2.3). But in contrast to an experiment design, we have to try to foresee changes within the study and set up fallback / alternate strategies to adapt the design to these changes. In the end the following elements should be defined (Runeson et al., 2012 [326]):

Rationale Why is the study done?

Purpose What is expected to be achieved with the study?

The case Overall, what is being studied?

Units of analysis In more detail (in case of a "multi case"-case study), what is being studied?

Theory What is the theoretical frame of reference?

Research questions What knowledge will be sought or expected to be discovered?

Propositions What particular (causal) relationships are to be investigated?

Define concepts and measures How are entities and attributes being defined and measured?

Method of data collection How will the data be collected?

Methods of data analysis How will data be analyzed?

Case selection strategy How will the case (and units of analysis) be identified and selected?

Data selection strategy How will data be identified and selected? For example, who will be interviewed? What electronic data sources are available for use in the study? What non-electronic, naturally occurring data sources are available for use in the study?

Replication strategy Is the study intended to literally replicate a previous study, or theoretically replicate a previous study; or is there no intention to replicate?

Quality assurance, validity, and reliability How will the collected data be checked for quality? How will the analysis be checked for quality?

Preparation of data collection After the case study is designed, all the selected means which are used throughout the study have to be prepared. This includes the selection and preparing of the cases as well as the means for revealing and documenting the data. The means for data collection depend on the chosen data collection method. In case study design common methods are *interviews*, *focus groups*, *observation*, and *analyzing archival data*. Interviews can be unstructured, which means that we have to prepare guidelines containing guiding questions which should be answered throughout the interview. The interview itself does not follow any prescript, but a normal course of a conversation. On the other side of the scale, we have structured interviews for which we have to prepare a catalog of closed questions (answers are predefined). The interviewee answers these questions and there is not necessarily an interviewer. In a semi-structured interview open and closed questions are mixed. Open questions are used for exploratory means, as the course of the interview is not predefined and the interviewees select the important aspects by themselves. This way, we can collect all important data, even unexpected data, but the analysis is hard as no interview is comparable to another interview. Closed questions instead allow a descriptive analysis of the data, and we can investigate specific causal relations we are interested in (explanatory purpose). A focus group is an unstructured interview, but with several interviewees in a group. This way we get an aggregated set of data, as the interviewees can respond to observations other interviewees state. When observing the participants of the study, we do not interview them but just look at their behavior. Hence, we have to prepare observation guidelines to make clear on which behavior we have to focus, and data observation sheets which structure the observations. Analyzing archival data is also kind of an observation in which we do not observe the participants directly, but we are looking at all the artifacts produced throughout the experiment. The different data collection methods can be combined.

Collection of evidence and analysis of collected data Then we have to conduct the study. The difference to an experiment is that we might respond to unexpected behaviors, inappropriate data collection methods, and so on. Hence, while collecting the evidence, we also have to reflect on the course of the study. Therefore, the analysis of the collected data is conducted in parallel to the collection of the data. Only this way we are able to detect, and then respond to, unexpected behavior, unexpected results, or flaws within the case study design. The process of data analysis starts with the codification of the collected data. Nevertheless, we have qualitative data, it has to be structured and classified in a way to allow any reasoning. Based on the codified data, we try to formulate hypotheses, which we constantly check against the current state of the data set. Those hypotheses which remain valid until the end of the study are then used to refine findings. Additionally, we can try to generalize the findings. Methods for analysis of qualitative data are, for example, constant comparison, or cross-case analysis.

Reporting Of course, the results of the case study have to be reported. In case study research, it often happens that we have to deal with different audiences. The reason is that the research is executed in a real context, for example, in a company. Hence, this company is interested in all observations which are relevant for it. But this is a very different audience than a scientific community. Not only that they expect to have the report in a different form, they are also often interested in different observations. This has to be clear from the beginning, before collecting evidence and observations, but also has to be considered when preparing the reports.

3.2.4.2. Action Research

> Action researchers are among those who assume that complex social systems cannot be reduced for a meaningful study[7]. They believe that human organizations, as a context that interacts with information technologies, can be understood as whole

[7]Note that Baskerville uses the term study as synonym for experiment.

> entities. ... The fundamental contention of the action researcher is that complex social processes can be studied best by introducing changes into these processes and observing the effects of these changes. ... The key assumptions of the action researcher [*are that*]
>
> - (1) social settings cannot be reduced for a study, and
> - (2) action brings understanding
>
> ... (Baskerville, 1999 [32])

These statements of Baskerville (1999 [32]) highlight the reasons for doing action research and the two main characteristics of action research. The first characteristic is that the object of study (for example an artifact or treatment) is introduced into a real setting. While this characteristic is also true for a field experiment, the second one differentiates action research from a field experiment, because in a field experiment the researchers do not directly take part in the studied process, while in action research this participation is considered to be crucial to understand the process and the effect of the object of study. Wieringa (2014 [388]), Wieringa and Moralı (2012 [389]), Stolen and Solheim (2007 [362]), and Baskerville (1999 [32]) also stress one further difference: In a field experiment, we are only interested in the effect introduced by the object of study and not necessarily in improving the process it is introduced to in the end. In contrast, in action research one direct goal is to achieve an improvement in the end. Hence, action research is a dualism of solving an actual problem, for example, a company has, and conducting research.

The origin of action research is in the field of social science and was introduced by Lewin (1947 [238]). Since this time, action research has been controversially discussed in science. Action research lacks some attributes which are widely acknowledged to be important for research: reductionism, repeatability and refutation (Checkland, 1981 [99]; Lau, 1999 [231]). In action research a case is not reduced to a minimum of variables which are necessary to study a treatment (reductionism). Hence, there might be always undiscovered relations and effects, which impact our results and thus flawing our conclusions. Additionally, each action research case is unique and therefore other researchers can not repeat it. But this is important to generalize results. And the data of action research is always qualitative and influenced by the subjective view of the conducting researchers. Hence, for other researchers it is hard to refute the point of view of the conducting researchers take and the conclusion they draw. As a result, many researcher see action research "... as social inquiry rather than social science." (Baskerville, 1999 [32]). Within action research it is a significant problem to balance between consulting the client and conducting research (Baskerville, 1999 [32]; Lau, 1999 [231]). But on the other hand, the same topics can be seen as argument im favor of conducting action research. To reduce a study always embodies the threat of removing important influences, and this threat is likely to happen in a complex, social setting. Hence, using reductionism for a setting in which the influential factors and their relations are unknown or hard to predict does not result in reliable results but in jeopardizing the chance of gaining useful insights. Within action research, we at least have the chance to discover unknown influential factors. Even when we do not discover them explicitly, they are part of the results. The repeatability of one unique case might not be that important or even desirable, because when in the "real world" cases do not repeat, conclusions drawn from repeating one specific case do not improve the generalizability. In this situation it is even desirable to have different cases which reflect the reality and allow to draw conclusions by comparing the cases. The refutation remains a problem in action research, but currently it seems to be best practice to trust other researchers as even for experiments which are reported in papers, the data sets derived from the experiment are mostly not made public. The author of this thesis is aware that this is not an argument in favor of action research,

but an indication for a problem in scientific practice in the software engineering community. But as long as this practice is accepted, the refutation is not a valid argument against action research. One strong argument for action research is the fact that action research creates scientific results which have relevance in practice and are more likely to impact practice (Wieringa and Moralı, 2012 [389]; Dickens and Watkins, 1999 [118]). Keeping in mind the drawbacks of action research, action research is still acknowledged as valid research strategy (Baskerville, 1999 [32]; Wieringa and Moralı, 2012 [389]; Stolen and Solheim, 2007 [362]; Wohlin et al., 2014 [393]).

Within action research we can distinguish two settings. One in which we are asked to solve a problem for which we do not already have a treatment. Hence, we start the design cycle right from the beginning. In the other setting we already have a treatment and apply it for cases which fit to the problem the treatment is made for. Hence, in this setting we are only validating our treatment we designed on our own, while in the first setting the client might be already involved in designing the treatment.

For both situations there are appropriate processes like the canonical action research process (Davison, Martinsons, and Kock, 2004 [113]) for the former situation, and the technical research process (Wieringa and Moralı, 2012 [389]) for the latter one. Both are refinements of the general process as described by Baskerville (1999 [32]). All these processes are like other research processes but stress the importance of the collaboration with the client, and provide guidelines and principles to integrate the researcher team with the client side team. We will not go into detail on these processes as they do not differ much from the general case study process and are also reflected in the design sciences process by Wieringa (2014 [388]) in general and the design cycle in particular (see Section 3.1.3.1[8]).

3.2.5. Grounded Theory

Grounded theory was introduced by Glaser and Strauss (1967 [158]) as a new way of theory building. Their fundamental assumption is that a theory can only emerge from data describing the reality. This was in contrast to the research at that time in which theories were constructed in mind using reasoning and logic. Afterward, these theories were tested in reality. Glaser and Strauss (1967 [158]) argue that this way not only much effort was spent to validate theories which were not connected to reality and it was impossible to validate them empirically, but also that theories which were already inherent parts of existing data did not emerge, because researchers did not use existing data as source for theory building. Additionally, they observed that sticking to the initial theory and not rethinking it while obtaining the data to validate it led to situations in which the correct theory was just a minor change away, but remained undiscovered even though the data strongly indicated the change. Hence, Glaser and Strauss (1967 [158]) formed grounded theory as a research process in which the data is a first class citizen and the theory is subject to change and emerges from the data.

Grounded Theory as such is a high level process accompanied with some guidelines for building a theory from data. Nowadays there are different implementations of grounded theory. This process is proposed to be a spiral process in which the main activities are repeated whenever new data is available, and new data is obtained on basis of the already known data. The process consists of the steps *data collection*, *data codification*, and *theory building*. Glaser and Strauss (1967 [158]) argue that researchers do not need any preliminary knowledge of the field before starting to collect data. Moreover, they see preliminary knowledge as a threat as the theory shall emerge from the data and not from the expectations of the researchers.

Data Collection For Glaser and Strauss (1967 [158]) everything is data: documents, interviews, and so forth. Hence, the method to collect the data depends on the type of data collected. Note that methods which already require a pre-structuring of the data are not recommended

[8] Page 45

to be used at least in the early iterations in which we are looking for new theories rather than stabilizing them. For example, closed questions within an interview already require knowledge about the possible answers and restrict the results. But in grounded theory the researcher should not restrict the data collection in this way.

Data Codification The codification of data happens in different phases. In the first iterations, we have to conduct *open coding*. Within open coding we try to identify reoccurring terms and relations between them. We explicitly do not consider already found codes[9] when we analyze new data. After the found codes seem to stabilize, we can additionally conduct *axial coding*. Within axial coding, we try to aggregate codes of the same kind, relate codes to each other, and so forth. Hence, we aim at forming a network of codes and generalize specific codes to more general codes. In the late phase, we apply *selective coding*. In this activity, we try to identify the *core codes* which can be found in each piece of data, and which might be the main parts of a theory. In the further iterations we are focusing on these core codes and only try to find and validate them in newly collected data. An important part of the codification is the *memo writing*. Thus, it is important to store the reasoning why something was identified as a code, why codes were aggregated, and so forth. Especially, when going from one level of coding to the next, this information is considered to be crucial by Glaser and Strauss (1967 [158]). For example, when selecting codes, the researchers should revisit the initial reason why a code was coded while doing open coding and how it was selected from the original data.

Theory Building Finally, we have to reason about the found core codes, how they fit together and what are the hypotheses why these are the core codes and how they are related. Again, the memos are of great importance, as they already contain a good amount of reasoning, and statements about relations between codes.

3.2.6. Literature Review

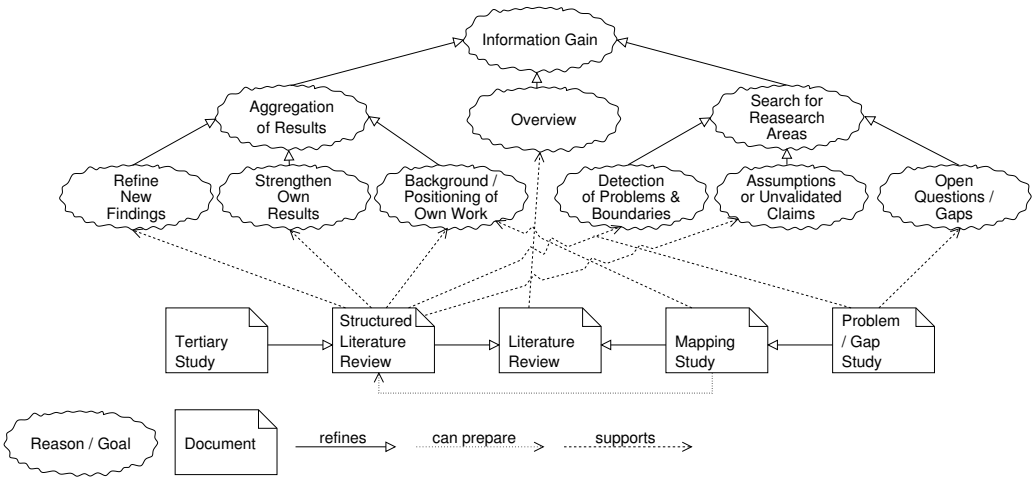


Figure 3.7.: *Types of literature researches and reasons and goals to perform them (Kitchenham, 2004 [214]; Kitchenham et al., 2007 [216]; Kitchenham et al., 2010 [217]; Kitchenham et al., 2010 [218]; Budgen et al., 2011 [81]; Brereton et al., 2007 [72]; Budgen et al., 2008 [80]; Petersen et al., 2008 [312])*

The following discussion on literature reviews is based on Beckers, Eicker, Faßbender, Heisel,

[9] Glaser and Strauss (1967 [158]) uses the term code for every observed phenomenon and construct we find in the data. This might be a term, but also a class of terms, or specific relations between terms. A code might be also build from different smaller codes.

Schmidt, and Schwittek (2012 [40])[10]. One kind of sample survey which is based on existing publications as knowledge source is a literature review. For the area of software engineering, Kitchenham was one of the first who described a structured literature review process (Kitchenham, 2004 [214]). Over the years this inital process was extended and improved by Kitchenham herself and others (Kitchenham, 2004 [214]; Kitchenham et al., 2007 [216]; Kitchenham et al., 2010 [217]; Kitchenham et al., 2010 [218]; Budgen et al., 2011 [81]; Brereton et al., 2007 [72]; Budgen et al., 2008 [80]; Petersen et al., 2008 [312]). Today it is the best practice approach and standard for undertaking literature reviews in the field of software engineering.

There are several reasons and goals why researchers might want to perform a literature review. And there are also different types of literature reviews, which can serve the different goals. Fig. 3.7 shows a condensed view of findings and statements from different publications in the field of systematic literature reviews in software engineering (Kitchenham, 2004 [214]; Kitchenham et al., 2007 [216]; Kitchenham et al., 2010 [217]; Kitchenham et al., 2010 [218]; Budgen et al., 2011 [81]; Brereton et al., 2007 [72]; Budgen et al., 2008 [80]; Petersen et al., 2008 [312]). The overall reason to do any kind of literature research is *Information Gain*. This top-level goal can be refined into the goal to get a mere *Overview* without a specific motivation (Kitchenham, 2004 [214]). This overview is not obtained for special reasons and the requirements for a literature review regarding completeness and quality are quite unspecific. In contrast, the goals of *Aggregation of Results* and *Search for Research Areas* have a well-founded motivation. Moreover, they lead to clear requirements for the quality, completeness and structure of a literature review.

When aggregating results, one might want to *Refine New Findings* based an the aggregated data (Kitchenham, 2004 [214]; Kitchenham et al., 2007 [216]). This goal is often the motivation when empirical studies are aggregated. Aggregating small datasets to a big one can lead to new insights. Or the findings and data of other publications are used to *Strengthen Own Results*, for example when the own dataset obtained is too small to allow statistically significant conclusions (Kitchenham, 2004 [214]; Kitchenham et al., 2007 [216]). A last reason for aggregation is to give a *Background / Positioning of Own Work* (Kitchenham et al., 2010 [218]; Budgen et al., 2011 [81]; Budgen et al., 2008 [80]; Petersen et al., 2008 [312]) and obtain a comprehensive background and rationale for the own research and to connect the results to the related work.

When searching for research areas, a *Detection of Problems & Boundaries* (Kitchenham, 2004 [214]; Kitchenham et al., 2007 [216]; Budgen et al., 2011 [81]; Brereton et al., 2007 [72]) of a certain method or set of methods can be the goal. The result then allows researchers to define where research is needed to improve (an) existing method(s). Another option is to search for *Assumptions or Unvalidated claims* (Kitchenham, 2004 [214]; Kitchenham et al., 2007 [216]; Budgen et al., 2011 [81]; Brereton et al., 2007 [72]). (In)Validating such assumptions and claims helps to improve the understanding of the research field for which they were stated. These two sub-goals aim at finding immature research areas and improve them with further research. In contrast, finding *Open Questions and Gaps* aims at research fields where no publications about solutions exist (Kitchenham et al., 2007 [216]; Budgen et al., 2011 [81]; Petersen et al., 2008 [312]). Such an area is a research gap, and often only an incomplete problem description for this gap exists. For example, several publications state the same question they did not tackle in their work and which is not directly related to this work, but which is a problem for future work.

All types of *Literature Review*s support the goal of obtaining an overview. The quality of the overview differs in how structured and planned the literature review is performed. A special type is the *Structured Literature Review* (SLR) (Kitchenham, 2004 [214]; Kitchenham et al., 2007 [216]; Kitchenham et al., 2010 [217]; Brereton et al., 2007 [72]). An SLR is a comprehensive literature review considering a specific research question. The SLR is obtained

[10]The discussion on literature reviews is a contribution of the author

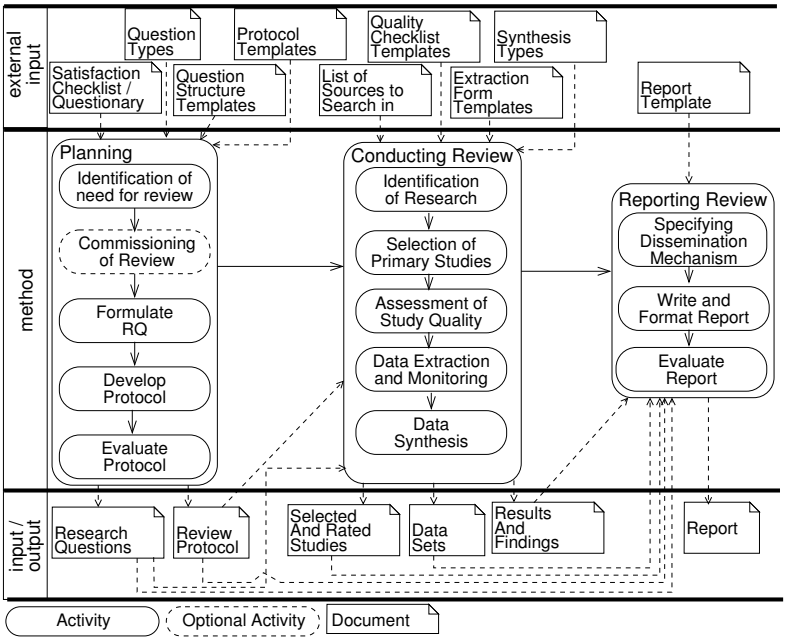


Figure 3.8.: *Process proposed by Kitchenham for undertaking a SLR (Kitchenham, 2004 [214]; Kitchenham et al., 2007 [216]; Brereton et al., 2007 [72]; Kitchenham et al., 2010 [217])*

in a planned and structured way. Moreover the SLR report is structured itself in a certain way. Kitchenham's method to perform an SLR was developed to find empirical primary studies considering a specific question and to aggregate the data in the first place. This review type supports all kinds of aggregation goals. Additionally, it turned out later that SLRs also make it possible for researchers to find immature research areas (Budgen et al., 2011 [81]; Brereton et al., 2007 [72]). For this kind of goal, Kitchenham's method has to be adapted slightly as the data to be extracted is no longer the empirical data itself but the findings based on this data. A special kind of the SLR is the *Tertiary Study* (Kitchenham et al., 2007 [216]; Kitchenham et al., 2010 [218]). Such a study aggregates the results of other SLRs, hence it relies on secondary studies. Another type of literature review is the *Mapping Study* (Kitchenham et al., 2007 [216]; Budgen et al., 2011 [81]; Budgen et al., 2008 [80]; Petersen et al., 2008 [312]). Here it is not the aim to extract any data, but to map studies to research fields or problems. In this way a good background can be established. This kind of study can be the first step to define the boundaries of an SLR. When also the problems and gaps discussed in the studies are obtained while doing the mapping, a *Problem / Gap Study* as a special kind of mapping study is the result (Kitchenham et al., 2007 [216]; Petersen et al., 2008 [312]; Budgen et al., 2008 [80]). This kind of study serves to find real gaps or to find immature methods. For assessing immaturity, a gap study should be combined with an SLR.

Kitchenham et al. also propose a process to conduct SLRs (Kitchenham, 2004 [214]; Kitchenham et al., 2007 [216]; Kitchenham et al., 2010 [217]; Brereton et al., 2007 [72]). It is shown in Fig. 3.8. This process can also be used for mapping studies with some slight adaptions (Kitchenham et al., 2007 [216]; Budgen et al., 2011 [81]; Petersen et al., 2008 [312]). The process is split up in three major phases. First the *Planning* phase takes place, followed by the *Conducting Review* phase, and finally the *Reporting Review* phase ends the review process.

The *Planning* starts with an *Identification of need for review* step, in which it is asked what the needs to be covered by an SLR are and if they substantiate conducting an SLR. The next step is optional as in *Commissioning of Review* the SLR is tendered to other research groups. This only happens if the group interested in the results is not willing to conduct the review themselves. The first real step towards an SLR is to *Formulate RQ* (Research Questions). The

research questions are the core of an SLR. All later decisions and results are checked against the RQ later on. In *Develop Protocol* the review itself is planned. The central aim of this step is to establish a common understanding of tasks and related documentation between the conducting researchers. The initial protocol is refined later on and also serves as a documentation of all steps executed. The step *Evaluate Protocol* is performed to detect misunderstandings, ambiguities, and insufficient definitions. An evaluation can be a test run on a small set of documents or just a recheck by SLR experts. To ease the planning there are predefined *Satisfaction Checklists / Questionaries*, *Question Types*, *Question Structure Templates*, and *Protocol Templates*. For *Question Structure Templates*, Kitchenham and Chaters propose to use the PICOC criteria framework to structure research questions (Kitchenham et al., 2007 [216]). PICOC stands for the criteria *population*, e.g. application area or specific groups of people, *intervention*, e.g. the method which is of interest, *comparison*, e.g. the benchmark, *outcomes*, what is the improvement to be shown, and *context*, a description of the setting in which the comparison takes place. All these documents serve as an input and guide for certain planning steps. The result of the planning phase are the *Research Questions* and the *Review Protocol*. They serve as input to the *Conducting Review* phase.

The review starts with the *Identification of Research*, which results in a set of studies which might be relevant. According to defined inclusion and exclusion criteria the step *Selection of Primary Studies* is performed. The selected studies are then rated in the step *Assessment of Study Quality*. For those studies with a satisfying quality level the data contained in the studies is extracted in the step *Data Extraction and Monitoring*. Afterwards the *Data Synthesis* is performed. The input to this phase are a *List of Sources to Search In*, *Quality Checklist Templates*, *Extraction Form Templates*, and *Synthesis Types*. Outputs produced in the conducting review phase are the *Selected And Rated Studies*, the *Data Sets* extracted form these studies, and the *Results and Findings* of the data synthesis.

All previously generated outputs serve as an input for the last phase *Reporting Review*. As external input, *Report Templates* are given. Based on the inputs, the step *Specifying Dissemination Mechanism* is executed, where one decides how to spread the result and in which form. Then the report is actually written in the step *Write and Format Report*. As a last activity an *Evaluate Report* step is performed. An evaluation can be a discussion with experts or a peer review when submitting the report to a journal or conference. The *Report* is the output of the entire SLR process.

3.3. Conclusion

In this chapter we have

- discussed the matter of software engineering research as design science and introduced the design and the empirical cycle of design science.
- given a brief overview on strategies, processes and methods which can be used within the two cycles.
- aggregated different sources about empirical research strategies and derived some information which might be helpful when selecting an appropriate strategy.
- surveyed works on structure literature review and derived different types of literature reviews and the goals they fulfill as well as a process for conducting a structured literature review.

The methods discussed in this Chapter are of particular importance to understand the validation and results presented in this thesis. In the next chapter (Chapter 4) we will discuss the relation between this chapter and the overall work.

Chapter 4

Used Scientific Methodology

In general, the results presented in this thesis were designed or derived following the ideas of design science as described in Section 3.1[1,2]. Hence, for all artifacts presented in this thesis, we iterated the design cycle as well as the empirical cycle. In the following, we will briefly highlight the methods actually used for conducting these cycles and how they will be presented in the following chapters.

4.1. Design Processes and Methods

As already mentioned in Section 3.2.1[3], for designing a new artifact one should combine the three different views (inspirational, structural, and situational), so did we. To describe the actual course of the design cycle, the processes, and especially the methods used for each artifact would exceed any reasonable amount of pages. Hence, we will only touch some important reoccurring themes in the following.

For the inspirational view, brain storming and mind mapping were the dominant methods used. For changing places, the main means was hiking and running as the author is a keen hiker and runner. Additionally, some piece of paper was always carried along to sketch new ideas whenever an illumination happened. But all in all, many methods and means were used, and it is hard to tell which of them worked, because an illumination triggered by a method can happen even weeks after using the method.

For the structural view, the central methods were small literature reviews. Small in this context means, that the literature reviews were not a full fledged SLRs as proposed by Kitchenham, but a very focused ones with a limited search depth. The topic of such a review was to understand the problem and its parts, or to find existing solutions. For structuring and combining solutions the main methods used were relevance trees, and morphological boxes. But all in all, the structural view was mostly taken for technology decisions. For all other decisions, the solution space was just to large to iterate it in a structured way. The structured methods only provided some first ideas from a limited part of the solution field to start with. These ideas found in a structured way were used afterward when applying to the other views.

The situational view was taken using different methods. One of them were research colloquia, which took place on regular basis in our working group. Such a colloquim was useful to discuss first ideas or even just for examining a problem. For discussing and getting new ideas regarding already elaborated artifacts which were almost paper ready, there were so called writers work-

[1] Page 41

[2] Note that Section 3.1 is mainly based on Wieringa (2014 [388]), which was published in 2014. Hence, most of the results presented in this thesis and the methods to obtain them were not directly influenced by this work. But they were influenced by the prior art on which Wieringa (2014 [388]) is based on such as Hevner et al. (2004 [184]), and Wieringa and Moralı (2012 [389])

[3] Page 50

shops[4] conducted by the working group on a regular basis. In such a workshop, the author does not actively take part, but only listens to what others think about a paper he / she submitted to the workshop. This serves two purposes. First, identifying where a paper transports the message insufficiently (which is not of importance for the design cycle itself, but for the reporting and dissemination), and more importantly to get new views on the artifacts presented and ideas for improving the artifact without influencing the discussions and brainstorming by the others. Of course, the standard means such as visiting, for example, conferences to talk to other researchers, or even collaborations for designing an artifact together with others were taken. Overall, the situational view turned out to be very important, as the solution space was vast and so were the different possible views on the problems considered in this thesis. Hence, combining the knowledge and ideas of different researchers was indispensable.

4.2. Empirical Methods

In the following, we explain which kinds of empirical methods were used in the context of this thesis. We also highlight which parts of this thesis rely on the information obtained with the different empirical methods. First, we discuss the experiments (Section 4.2.1), and the case studies (Section 4.2.2) conducted to validate the artifacts presented in this thesis. Next, we discuss the matter of grounded theory and how it inspired some of parts of our work (Section 4.2.3). Last, we explain how we got an overview about a research field and how we identified existing problems and gaps within this field using literature reviews (Section 4.2.4).

4.2.1. Experiments

Artifact	Chapter	Feasibility Experiment	Experimental Simulation
Context Patterns	Chapter 10 (Page 181)	2	2
UPROM2PF (Process Elicitation)	Section 11.2 (Page 206)		1
PresSuRE (Security Requirements Elicitation)	Chapter 13 (Page 221)		2
Compliance Requirements Elicitation	Chapter 17 (Page 307)	1	1
Interaction Detection	Section 18.2 (Page 323)	1	1
Generation of Alternatives	Section 18.3 (Page 327)	1	
Requirements Valuation	Chapter 19 (Page 333)	1	
Requirements Optimization	Chapter 20 (Page 341)	1	

Table 4.1.: *Overview of Conducted Experiments*

For some of the artifacts presented in this work, experiments were conducted. All of the experiments had an exploratory character. Hence, we tried to understand which variables influence the applicability and feasibility of the designed artifacts. Here, we distinguish two types of experiments. First, the so called feasibility experiments. They were conducted by us to get a first understanding of the designed artifacts and how it works. For such an experiment, we used crafted desktop examples or examples derived from a real case to which the artifact was applied by ourselves. In most cases, the used examples were especially crafted to test several aspects of the artifact. Thus, a feasibility experiment is completely artificial. The expected result of such an experiment is an increase in experience with and initial knowledge about the artifact and if its application is sufficient and feasible for a problem at hand. No benchmark in form of, for example, known results from real life cases exist for assessing the results. In consequence, the results are of qualitative nature which have a complete lack of generality and realism. But still, they are useful to conduct the treatment validation activity in the early design cycle iterations for an artifact. As already discussed, one should start with experiments which have a low scale and risk and, while the knowledge about the artifact increases, conduct subsequent

[4] http://hillside.net/conferences/plop/235-how-to-hold-a-writers-workshop

experiments which are more complex (see Section 3.2.2[5]). Hence, as a starting point such a feasibility experiment is sufficient.

All other experiments were experimental simulations as defined in Section 3.2.2[6]. Here we used real life cases for which already the results were known. For example, for the voting system the requirements are publicly available as well as the court judgment. Hence, we have realistic inputs and also a benchmark we can compare to. The artifact was applied by a team reflecting the team which would apply the artifact in a real case. Afterward, the results of the artifact application were compared to the benchmark. This way we obtained more realistic results for assessing an artifact.

Table 4.1 shows the experiments conducted in the context of this thesis. In the first column the table shows the artifacts, in the second column the chapter in which the experiments will be discussed in detail, and in the third and fourth column which kind of experiments were conducted and how many. Note that the number for the context patterns might be misleading as there are several context patterns which are single artifacts. Hence, the experiments are distributed over the different context patterns. The experimental simulations themselves are reported in a brief form using the template proposed by Runeson et al. (2012 [326]) (see Section 3.2.4.1[7]) in Appendix A.1[8]. The experiments and the according results are discussed in detail in the chapters shown in Table 4.1.

4.2.2. Case Studies

Artifact	Chapter	Case Study	Action Research
Context Patterns	Chapter 10 (Page 181)	2	2
Compliance Requirements Elicitation	Chapter 17 (Page 307)		1

Table 4.2.: *Overview of Conducted Case Studies*

Some of the artifacts presented in this work were also validated using case studies and action research as defined in Section 3.2.4[9]. Table 4.2 shows the studies conducted in the context of this thesis. In the first column the table shows the artifacts, in the second column the chapter in which the case studies will be discussed in detail, and in the third and fourth column whether a case study or action research were conducted and how many different studies were conducted. The case studies and action research themselves are reported in a brief form using the template proposed by Runeson et al. (2012 [326]) (see Section 3.2.4.1[10]) in Appendix A.2[11]. The experiments and the according results are discussed in detail in the chapters shown in Table 4.2.

4.2.3. Grounded Theory

Grounded theory as such was not directly applied within this work. But the way of obtaining context patterns from existing information was inspired by this method. Note that when we started to work on the context patterns, we were not aware of grounded theory and its potential for our work. Hence, the process described in Chapter 7[12] emerged from our own work. Afterward, we compared grounded theory to this process. We found many similarities which gave

[5]Page 52
[6]Page 52
[7]Page 58
[8]Page 377
[9]Page 58
[10]Page 58
[11]Page 381
[12]Page 113

some scientific ground to our process and we also changed the process slightly to adapt better to important aspects of grounded theory. This way the process for deriving context patterns is an implementation of grounded theory specific for the purpose of our work.

In consequence, the context patterns themselves can be seen as theories, which emerged from the documents, interviews, and so forth, they are based on. In general, we argue that the pattern idea of patterns emerging from already existing solutions are in line with the idea of grounded theory, in which theories are hidden within the data describing the real world. Hence, grounded theory gives the pattern idea some additional scientific grounding.

4.2.4. Literature Review

For writing this thesis, literature reviews were one of the central tools. The goals of interest for conducting these literature reviews were getting an *overview*, find *open questions and gaps*, and *positioning of own work* (see Figure 3.7[13]). Hence, we conducted basic *literature reviews* as well as *problem / gap studies*. For each topic of interest we started with a literature review. In case the literature review revealed enough answers to our research question or enough solutions for our problem at hand, we stopped. If the overview provided by a literature review indicated that there might be some gaps, we extended the literature study to a problem / gap study.

For this thesis no structured literature review was conducted. As most of the reviews were conducted by the author alone, conducting a full fledged SLR was beyond feasibility. Hence, the results of the reviews are not sufficient to refine new findings or strengthen the results of the artifacts presented in this thesis. The reader should be aware of this limitation when interpreting the presented results.

4.2.4.1. Protocol

In the following section, we use the mapping study protocol template suggested by Kitchenham (2009 [215]) to describe our reviews. We only explain the commonalities between all the reviews conducted, while the details which are specific for each review can be found in Appendix A.3[14]. As already mentioned, we conducted two types of reviews, ordinary literature reviews and problem / gap studies. As a gap study extends a literature review, we will highlight the additional steps taken by *highlighting them in italics*.

4.2.4.1.1. Background According to Kitchenham (2009 [215]), in the background we have to

- explain why there is a need for a study on this topic,
- identify the topic that is to be 'scoped' in the study, and
- specify any research question that will be addressed.

As this is specific for each review, we can not explain any commonalities at this point.

4.2.4.1.2. Search Strategy For this thesis, we decided to use only a manual search. There are two reasons for this decision. First of all, for conducting an automated search one has to know the specific terms related to the topic in question. But many of the topics in question within this thesis were (partly) unknown to the author in the beginning. Within a manual search researchers can easily reflect on the new insights gained, and adjust their search accordingly. Additionally, a researcher is able to judge whether a term might be related to the topic in question even though the term was not explicitly specified and known beforehand. Hence, whenever exploring

[13] Page 63
[14] Page 387

a new field, a manual search is favorable. Another reason we do not want to conceal is that from our experience with automated searches an automated search bears much more effort to be spent than a manual search. Because when executing an automated search many false positives are generated, which still have to be investigated. The false positives are hard to remove, as a refinement of search terms always bears the risk of excluding relevant works. Hence, it was reasonable and sufficient for us to conduct a manual search only.

For selecting the initial conferences and journals, we searched the CORE and ERA rankings[15] for A and B level journals in the fields of software engineering in general and requirements engineering in specific. This way we got a list of conferences and journals which served as baseline for each literature review (see Table A.1[16] in the appendix).

We also extended this baseline list for each topic in question. First, by manually searching for topic specific workshops at the conferences of the baseline list. Second, by searching for topic specific conferences and workshops using Google. The topics, according search strings, and results are shown in Table A.2[17] in the appendix. Of course, when searching the topic specific list, the other topics were also in mind. Hence, whenever a suitable paper was found for one of the topics in mind, it was added to the results even when the source conference / workshop was added for another topic.

The manual search was conducted for the time frame starting from 2009 up to now, because for getting an overview, the most recent works were sufficient. Of course, the time span searched increased overtime, as the year 2009 was fixed, but each iteration of a review took place in a later year than the previous iterations.

For a normal literature review, also the related work sections for papers at hand were read, and some references were followed but no systematic snowballing was applied.

For a gap study, works from a limited time span are not sufficient to derive any conclusions. Hence, we used backward snowballing. The activity of analyzing the bibliography of a paper already identified as relevant to find further candidates is called backward snowballing. According to Jalali and Wohlin (2012 [203]), results derived from a manual search combined with backward snowballing are comparable to results from an automated search. Hence, we get reliable results this way.

The search was justified over the years by talking to other researchers, and relevant papers on the topic they might know. Whenever important papers came up, we checked whether the paper was already identified. In case it was not already identified, we investigated the reasons. Sometimes this triggered a further iteration, as we might have overlooked some important terms or even conferences or workshops. This way, the searches were adjusted over time, and up to now, the author is not aware of any important work on the topics which is not part of the relevant works identified by the reviews.

4.2.4.1.3. Selection Criteria The only inclusion criteria used was the relevance for the topic at hand. The relevance was judged by the author. *First*, only *title and abstract* were used to include each paper which might be of relevance. Here, the author selected very defensively. Every paper was included for which there was a slight chance that the paper was relevant. Hence, in a *second step*, the author read the *introduction, discussion, and conclusion* of each paper. Some papers could be excluded this way. The rest of the papers were read in full length and included in the results.

No further quality evaluation was conducted. The reason is, when doing a review alone, one cannot assure the quality of the quality judgments by, for example, comparing different

[15] http://www.core.edu.au/
[16] Page 387
[17] Page 388

judgments for the same paper, which is indispensable for a reliable quality judgment (Kitchenham, 2004 [214]). Additionally, checking quality criteria bears additional effort, which is not reasonable for one person alone (Kitchenham, 2004 [214]). Moreover, for a mapping study skipping the quality evaluation is acceptable as we do not aggregate data from the different sources to a new data set, but add statements from each source to the discussions within this work separately. Hence, the reader can judge the reliability of a source him-/ herself, even though the initial selection of journals, conferences, and workshops already implies a certain degree of quality.

4.2.4.1.4. Data Extraction For the literature reviews no particular data extraction process was defined. The reviews only served the purpose of getting an overview. Of course, the related work sections are based on the results of the literature reviews, and sometimes also existing solutions were integrated into the designed artifacts. Hence, the description of the related work and the used solutions are kind of a data extraction, but a rather limited one because the extraction does not follow a structured form.

For the problem and gap studies, the data extraction followed some structure. Each paper was analyzed for statements which highlight the importance of a certain topic, activity, and so forth, or which coined a particular problem. For each of these statements it was also collected if the statement applies for all fields of IT, for software engineering in general, or for requirements engineering in specific.

4.2.4.1.5. Synthesis For the literature review no synthesis happened. *For the problem and gap studies, the statements collected were compared to each other and grouped by the topic they highlight or the problem they describe. For some of the gap studies, some more information was collected, but as those data is study specific, it will not be described at this point. Afterward, descriptive statistics were used to analyze the data.*

4.2.4.1.6. Study Limitations As an ordinary literature review does not claim to have any stakes in being complete or that the results are generalizable, there is no need for discussing threats to validity.

For each problem and gap study, the threats to validity have to be discussed. In general, we can say that we followed the best practices in the field of literature reviews. This way we tried to minimize the threats to validity. But still, there are some important threats to validity to name[18] *and we identified two major threats to validity.*

First, when only one researcher is doing a study, the threat that the researcher biases the results (internal threat to validity) by taking a subjective point of view is staggering. There is no second opinion to check against, and means to calculate agreement indicators and methods to settle disagreement are not applicable. As long as it is not possible to collaborate with at least one more researcher there is no direct countermeasure we can take against this threat. But as were are only doing gap studies, the level of aggregation is rather low. Providing the direct references to the papers where statements originate from enables the reader to check the validity of conclusions drawn. Furthermore, all other data presented within this thesis is directly provided in the appendix and linked to the original sources. Hence, the reader is able to check the validity of conclusions. Additionally, all chapters in this thesis were discussed at least with one more researcher. Hence, some kind of peer review has happened.

The second important threat to validity (external threat to validity) is the completeness. From our point of view, this threat is minimized by the fact that the number of venues searched is

[18] We considered and checked the threats to validity as enumerated by Wohlin et al. (2014 [393])

quite big, and covers the most prominent venues in the field. Additionally, the backward snowballing without time limitations has shown to be suitable to ensure completeness (Jalali and Wohlin, 2012 [203]).

Other threats are not that significant due to the character of our studies. As we did not apply a quality selection, beside the quality based selection of venues, this step cannot bias the results. Also the coherence of data extraction decisions of different researchers is not of relevance, as only one researcher conducted them. As there is not a high level of aggregation, there are hardly any construct threats to validity which are concerned with the validity of metrics and measures used for aggregation.

4.2.4.2. Overview of Conducted Literature Reviews

Topic	Chapter	Literature Review	Problem & Gap Study
Decision Making in Requirements Engineering	Chapter 1 (Page 3) Chapter 19 (Page 333) Chapter 20 (Page 341)		X
Context Elicitation	Chapter 1 (Page 3) Chapter 6 (Page 81)		X
Patterns	Chapter 1 (Page 3) Chapter 6 (Page 81)		X
Goal Elicitation	Section 11.1 (Page 203)	X	
Process Elicitation	Section 11.2 (Page 206)		X
Security Requirements Elicitation	Chapter 13 (Page 221)	X	
Compliance Requirements Elicitation	Chapter 1 (Page 3) Chapter 16 (Page 281)		X
Interaction Detection	Section 18.2 (Page 323)	X	
Generation of Alternatives	Section 18.3 (Page 327)	X	
Requirements Valuation	Chapter 19 (Page 333)		X
Requirements Optimization	Chapter 1 (Page 3) Chapter 20 (Page 341)		X

Table 4.3.: *Overview of Conducted Literature Reviews*

Table 4.3 shows the literature reviews conducted in the context of this work. In the first column it shows the topic, in the second column the chapters in which the results are reported, and in the third and forth column which type of literature review was conducted for this topic.

4.3. Conclusion

In this chapter we connected the background on scientific methodologies as presented in Chapter 3[19] with the work presented in this thesis. This way, the reader might be able to judge the scientific grounding of this work, and to reason about the credibility of motivations given and problems described, as well as the reliability and sufficiency of the presented solutions within this work.

[19] Page 41

Chapter 5

Cases Used

In this chapter we introduce the use case of a media market which is used to sell media of different content providers to customers (Section 5.1). The media market case will serve as running example throughout the thesis. It is based on (Beckers, Faßbender, Heisel, and Meis, 2012 [41])[1]. Two further cases, which were used for validation purposes, are introduced in Section 5.2.

5.1. Running Example

For our running example, we have chosen a media publishing and retrieval setting. In this setting, there are customers who want to retrieve certain media. For example, this media can be a piece of software, a video or movie, a song, or an e-book. On the other side, we have various content providers. A small content provider may offer only one media type and only a small selection of media to choose from. In contrast, a big publisher usually offers all media types and a big selection of media.

The main problem in the relationship between customers and content providers is that on the one hand the customers prefer a uniform search and access interface and do not want browse a big number of different shops, with different access technologies, credentials, and so on. A second problem for customers is to oversee the whole market. Customers might not even know the right content provider for very special media. On the other hand, not all content providers are able or willing to set up and maintain a shop infrastructure with essential functionality such as billing.

The business idea of our example is to introduce a content aggregator as a mediator between customers and content providers. The aggregator collects the offers of different content providers. These offers are aggregated by the aggregator and then made available to the customers. The content aggregator also handles the payment by integrating banks into the business process.

The content aggregator decides to choose a SOA to realize its business, because of the dynamics in this setting. There is a huge number of providers. The aggregator wants to be able to find and integrate these providers at run-time. Moreover, the access for the content providers to the content aggregator should be as simple as possible. Therefore, services are reasonable. Services enable the providers to wrap their existing technologies and use standard protocols.

A second reason for using services is the fact that the content provider has no direct access to the devices the customers use to search for media. A device in this case can be, for example, a smartphone, a settop box, or a tablet PC. Hence, the content aggregator has to ensure that customers can find the solution the aggregator provides. Moreover, the content aggregator has to ensure that the technology used for providing the solution can be easily adapted for each device. Services are a good choice to achieve the goals of easy finding and integration of different devices and software platforms. In Section 2.1.6[2] we have provided more detailed arguments supporting

[1] The example was contributed by the author.
[2] Page 30

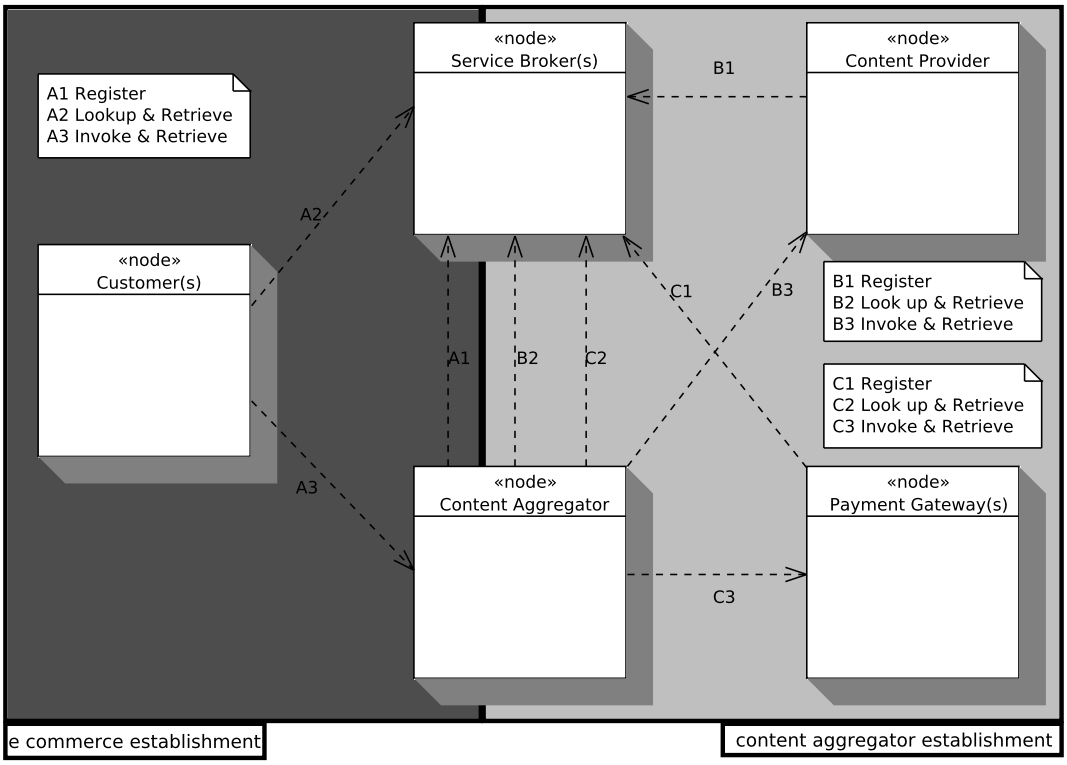


Figure 5.1.: *UML Deployment Diagram: SOA Scenario*
A=Business Case,B=Content Provider Integration,C=Payment Gateway Integration

this claim.

A third reason for using services is the needed scalability of the solution. Over time, the number of customers increases and therefore also the number of transactions which have to be handled by the aggregator as well as the content providers. Cloud providers offer scalable platforms for running services.

A last reason for using services would be the large number of different banks and online payment systems, which have to be integrated to fulfill the payment needs. But this requirement is already addressed by so-called *payment gateways*, which aggregate all banks and make payment functionalities available at a single point. These payment gateways offer their functionality also by services. At this point, we use the best practice already established in the market.

The resulting scenario is shown in Fig. 5.1. The figure shows the common service look up and invocation process. There are three different instances (A, B, C) of the Register, Look Up & Retrieve, and Invoke & Retrieve sequences, which are typically for a SOA setting, in our scenario. The content aggregator queries a service broker to find content providers (Arrow B2 in Fig. 5.1) and payment gateways (C2) to establish its business. Content providers (B1) and payment gateways (C1) registered themselves at the service brokers before. At run time, the content aggregator invokes the services of the providers (B3) and gateways (C3). Note that the set of gateways will be almost static and slowly evolving as there are not that many payment gateways, and establishing the needed service-level agreements (SLAs) and trust cannot be achieved on the fly. In contrast, the set of content providers can be different for each business process invocation. The market of providers is changing fast, and standard SLAs can be used, so service invocation can be done automatically. To realize the business case, content aggregators register their service at a service broker (A1). The customers are now able to find the aggregators (A2) and search for and retrieve (A3) the desired content.

5.2. Further Cases

The two real world cases which are used for validation of different parts of this work are a voting system (Section 5.2.1), and a Smart Metering System (Section 5.2.2). Section 5.2.1 is based on Faßbender and Heisel (2013 [133]). Section 5.2.2 is based on Alebrahim, Choppy, Faßbender, and Heisel (2014 [6]), and Faßbender, Heisel, and Meis (2014 [135]).

5.2.1. Voting System

Electronic voting enjoys an ever-increasing interest from governments as they seek to facilitate and simplify their election procedures. In several countries like Australia, Brazil, Belgium, Canada, India, UK, and Spain electronic voting systems are already in use (Kumar and Walia, 2011 [228]). In Germany there are also some activities on introducing such a voting system. By its very nature, the field of electronic voting is an interdisciplinary field in which legal and computer scientists work together. During the development of the first voting system used in Germany, this fact was neglected or inadequately considered. Hence, the federal constitutional court of Germany judged in 2009 that using this system for elections in 2005 was unconstitutional (Federal Constitutional Court of Germany, 2009 [140]).

A general problem description of this voting system and which functionality it has to provide was derived from Brehm (2012 [71]) and Volkamer (2009 [382]). The former work was conducted in the context of the ModIWa II project, while the latter work was elaborated in the context of a Common Criteria (CC) Profile (Volkamer and Vogt, 2008 [383]) for online voting systems. These documents were used for detailing the requirements and knowledge about the involved stakeholders and systems, and their relation to each other (see Appendix F.3[3] for the full results).

Brehm (2012 [71]) describes the reference process and the systems used within this process as follows. The **election authority** represented by an **election officer** starts the polling process and the **ballot** is sent to the **voters**. The needed information about the voters is obtained from an **electoral register**. The voter completes the ballot using his / her **voting device** and sends the ballot back. The incoming ballot is stored in a digital **ballot box**. When the polling phase is over the ballots contained in the digital ballot box are counted and the election results computed. Volkamer (2009 [382]) concretes the involved systems and their relation to each other. First of all, she establish a client server infrastructure. Moreover, she introduces the **tallying authority**, responsible for generating the voting results, the **registration authority**, responsible for maintaining the electoral register, and the **election observers**, who control the election itself. These authorities are distinct from the election authority described by Brehm [71]. The election observers, the tallying authorities, and the registration authorities use their own systems which are connected to the voting system. These systems have to be separated from the voting system itself, but are also part of the overall problem as they have to be developed in the context of the voting system. Hence, Volkamer (2009 [382]) gives a complete overview, in terms of involved systems and stakeholders, of the voting system, but does not describe functional requirements or the process in which the voting system is used.

The CC profile for online voting systems (Volkamer and Vogt, 2008 [383]) only deals with the polling phase. The preliminary election preparation and the tallying are not considered in detail. Hence, the profile only defines functional requirements regarding the usage of the system by the voters and election officers who represent the election authority. In total, the machine to be built is described in terms of 21 requirements by the CC profile. Later on, we have split some of the requirements for handling reasons. Besides the requirements themselves, the knowledge about the environment of the machine to be built is crucial for understanding the problem and

[3] Page 549

specifying the machine behavior later on. In total, the CC profile states 21 assumptions, 14 facts, and 35 definitions and designations.

5.2.2. Smart Meter

To use energy in an optimal way, smart grids make it possible to couple the generation, distribution, storage, and consumption of energy. Smart grids use information and communication technology (ICT), which allows for financial, informational, and electrical transactions.

To illustrate the validity and sufficiency of the different works which are part of this thesis, we use the real-life example of smart grids. We focus on the smart meter within a household. As sources for real functional and quality requirements, we consider diverse documents such as "Application Case Study: Smart Grid" and "Smart Grid Concrete Scenario" provided by the industrial partners of the EU project NESSoS[4], the "Protection Profile for the Gateway of a Smart Metering System" (BSI, 2011 [76]) provided by the German Federal Office for Information Security[5], and "Requirements of AMI (Advanced Multi-metering Infrastructure") (OPEN meter project, 2009 [297]) provided by the EU project OPEN meter[6].

We first define the terms specific to the smart grid domain: The **Gateway** represents the central communication unit in a *smart metering system*. It is responsible for collecting, processing, storing, and communicating *meter data*. **Meter data** refers to meter readings measured by the meter regarding consumption or production of a certain commodity. **Meter** represents the device that measures the consumption or production of a certain commodity and sends it to the gateway. An **Authorized external entity** could be a human or IT unit that communicates with the gateway from outside the gateway boundaries through a *Wide Area Network (WAN)*. The roles defined as external entities that interact with the gateway and the meter are *consumer, supplier, gateway operator, gateway administrator, ...*[7]. The **WAN (Wide Area Network)** provides the communication network that interconnects the gateway with the outside world. The **LMN (Local Metrological Network)** provides the communication network between the meter and the gateway. The **HAN (Home Area Network)** provides the communication network between the consumer and the gateway. The **LAN (Local Area Network)** provides the communication network that interconnects domestic equipment or metrological equipment[8]. The **Consumer** refers to the end user or producer of commodities (electricity, gas, water, or heat).

For the smart grid, different quality requirements have to be taken into account. Detailed information about the energy consumption of the consumers can reveal privacy-sensitive data about the persons staying in a house. Hence, we are concerned with privacy issues. A smart grid involves a wide range of data that should be treated in a secure way. Additionally, introducing new data interfaces to the grid (smart meters, collectors, and other smart devices) provides new entry points for attackers. Therefore, special attention should be paid to security concerns. The number of smart devices to be managed has a deep impact on the performance of the whole system. This makes performance of smart grids an important issue.

For applying our methods, we selected the 13 minimum uses cases from
OPEN meter project (2009 [297]), which embody 27 functional requirements in total. The requirements and according problem diagrams are shown in Appendix F.2[9].

[4] http://www.nessos-project.eu/
[5] www.bsi.bund.de
[6] http://www.openmeter.com/
[7] For the complete list of possible external entities see the protection profile [76]
[8] In protection profile, LAN is referred to as hypernym for LMN (Local Metrological Network) and HAN (Home Area Network).
[9] Page 549

Part II.

Understanding the Purpose

Chapter 6

A View on Context Elicitation and Pattern Languages

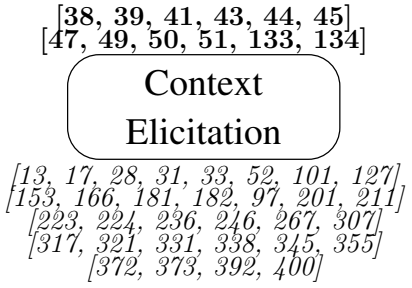


Figure 6.1.: *Context Elicitation*

The first step when eliciting a coherent set of requirements which satisfy all stakeholders is to collect all relevant entities of the environment of the system-to-be. We propose to use domain specific patterns, which help one to point out where to look for relevant information. This information is documented using graphical patterns, which are accompanied with textual templates. The documented information not only describes the relevant entities, but also captures their relation to each other and to the system-to-be. Entities can be stakeholders, assets, systems, and so forth. Figure 6.1 shows the works our pattern-based method for context elicitation is based on (**bold**) and the related work for the chapters on this matter (*italic*). In this chapter we will give a general introduction to the importance of context elicitation, existing problems in this field, and the idea of patterns and a pattern language for context elicitation. In Chapter 7[1], we will discuss some fundamentals for understanding our context patterns, a metamodel and process for deriving new context patterns. This chapter is followed by the catalog of context patterns which we have already derived (Chapter 8[2]). In the chapter after, Chapter 9[3], we will relate the different context patterns and form a pattern language. Chapter 10[4] will conclude the chapters on the matter of context elicitation. It contains the application of the relevant context patterns for our running example as well as a discussion about the validation for our context patterns.

6.1. Overview

We start this chapter by analyzing the importance of context elicitation for requirements engineering (Section 6.2). The insights presented in this section are based on different statements derived from research and studies on the matter. Together with a discussion of the existing problems in context elicitation in research as well as in practice, we will motivate our work on context

[1] Page 113
[2] Page 131
[3] Page 151
[4] Page 181

patterns for context elicitation. Next, we present and discuss our basic idea of using patterns for context elicitation, and provide some reasoning for using patterns for this matter (Section 6.3[5]). These two sections are based on a problem and gap study we have conducted in the field of context elicitation for requirements engineering. In Section 6.4[6], we discuss further results from the study, position our work in the existing body of knowledge in the field (Section 6.4.1[7]), and survey the existing related work on context elicitation (Section 6.4.2[8]) and describing patterns and pattern languages (Section 6.4.3[9]). Up to this section, the present results are novel and unpublished insights. Section 6.5[10] is based on (Beckers, Faßbender, and Heisel, 2014 [50])[11] and discusses the matter of patterns and pattern languages and what constitutes them. Before we conclude this chapter in Section 6.7[12], we provide a quick overview of the existing context patterns (Section 6.6[13]) as a basic overview helps understanding the following chapter.

6.2. Motivation

The notion of context in relation to software systems comes in different shades. Starting from a purely technical context which only includes existing components and systems (see, for example, a workshop summary of Tracz (1992 [370])), over the direct context which includes technical parts as well as people which directly interact with a system (for example, Jackson and Zave (1993 [201]), and Jackson (2001 [200]), to a broad view of context which also includes, for example, indirectly influenced stakeholders (for example, Curtis et al. (1988 [108])). For an elaborate definition of context see Section 2.1.5[14].

The discussion about considering the context of a system-to-be in software engineering and especially in requirements engineering is not a new one. Starting in the late 1980s (for example, the work of Curtis et al. (1988 [108])) where the context of a system appeared as a relevant factor for the success of software engineering, the topic was also considered as relevant part of the, at that time, emerging field of requirements engineering (for example, Jackson and Zave (1993 [201])) in the 1990s, and gets increasing attention in the 2000s and 2010s by (empirical) studies in the field of requirements engineering, which investigate the existing barriers to successful requirements engineering (for example, the studies of Damian, Helms, Kwan, Marczak, and Koelewijn (2013 [109]), Meth, Maedche, and Einoeder (2013 [268]), Niknafs and Berry (2012 [284]), Davis, Dieste, Hickey, Juristo, and Moreno (2006 [112]), and Coughlan and Macredie (2002 [105])). Hence, the topic of context and the related domain knowledge needed to understand it is known since the advent of requirements engineering.

But why is the context of a system-to-be and the knowledge about it regarded as that important for a successful development of a system-to-be? First of all, there are findings which indicate that aligning IT systems to the overall organizational and business goals enables organizations to outperform their competitors (Singh and Woo, 2009 [355]). But for developing an IT system which perfectly integrates into the needs of an organization, software engineers need detailed knowledge about the context of the system-to-be. This is also emphasized by Maalej and Ghaisas (2014 [248]) who state that

[5] Page 87
[6] Page 91
[7] Page 92
[8] Page 96
[9] Page 100
[10] Page 102
[11] The initial discussion on patterns and pattern languages was contributed by Kristian Beckers, but was revised by the author and evolved overtime to the current state as presented in this thesis.
[12] Page 111
[13] Page 107
[14] Page 30

> One main lesson learnt from this study is that nowadays software organizations do not only rely on the asset of the software, technology, and design knowledge, but also more and more on the domain knowledge in general and business rules in particular.

Additionally, also IT systems which are not bound to a specific organization have an increasing reliance on the environment (Cheng and Atlee, 2007 [100]), as IT systems are becoming an integrated part of many domains and aspects of our daily life. In general, it is acknowledged since the early beginnings of software engineering that

> Experts generally solve problems in their fields more effectively than novices because their well-structured, easily activated knowledge allows for efficient search of a solution space. Wiley (1998 [390])

As a result, domain understanding is regarded as key to successful system development (Offen, 2002 [292]; Coughlan and Macredie, 2002 [105]; Fuentes-Fernandez et al., 2010 [153]; Sadraei et al., 2007 [332]), which is seconded by a study as reported by Buchan and Hasana Ekadharmawan (2009 [78]), and Buchan et al. (2009 [79]) who state

> The collaborative development of a shared domain understanding between the client stakeholders and the software production team is crucial to the success of software development projects.

.

In particular, domain understanding is an important factor for successful requirements engineering. Again, there is a business aspect as software engineers are often working for differing clients in different business domains. Hence, Ugai and Aoyama (2009 [373]) state

> It is important to know business and data that relates to customers' needs so that we can elicit requirements. Put simply, to elicit requirements we need particular knowledge about our customers' businesses. Here, such knowledge is called "domain knowledge", and our customers' businesses are called a "problem domain".

Other researchers second this view that having a deeper understanding of a business domain (Osada, Ozawa, Kaiya, and Kaijiri, 2007 [300]) and, for example, building a domain model is a necessity in requirements engineering (Garcia Alcazar and Monzon, 2000 [155]; Sutcliffe and Sawyer, 2013 [363]). Hence, in the field of requirements engineering for business enabling software there is a need to understand the particular business domain in question.

But there are also more general reasons for the importance of context in requirements engineering. Pacheco and Garcia (2012 [303]) state that

> Improving requirements elicitation requires us to first understand the stakeholder identification (and selection) phase.

The importance of stakeholder identification and selection is seconded by other papers and studies (Azmeh et al., 2013 [28]; Hofmann and Lehner, 2001 [185]; Sadiq and Jain, 2014 [331]; Ballejos and Montagna, 2008 [31]; Alexander and Robertson, 2004 [13]). Indeed, stakeholder consideration as part of the context elicitation is developing to a topic on its own (we will ground this statement in Section 6.4.1[15]). Nevertheless, stakeholders are not the only important part of the context of a system-to-be (Jackson and Zave, 1993 [201]; Jackson, 2001 [200]; Tracz, 1992 [370]; Curtis et al., 1988 [108]). This is highlighted by Nuseibeh and Easterbrook (2000 [290]):

[15] Page 92

> One of the most important goals of elicitation is to find out what problem needs to be solved, and hence identify system boundaries. These boundaries define, at a high level, where the final delivered system will fit into the current operational environment. Identifying and agreeing a system's boundaries affects all subsequent elicitation efforts.

Hence, all parts of the context are of importance, as requirements engineering is a communication intense activity, and the discussion of requirements with different stakeholders requires a common domain understanding of all participants about the complete context of a system-to-be (Leonardi et al., 2010 [236]; Annervaz et al., 2013 [17]; Liu et al., 2010 [244]).

Neglecting the importance of context elicitation and subsequently ignoring the context and the related important domain knowledge has several negative consequences as observed by different researchers. Curtis et al. (1988 [108]) state that missing a shared understanding of the context is one key factor for communication flaws. Curtis et al. (1988 [108]) also observed that

> The deep application-specific knowledge required to successfully build most large, complex systems was thinly spread through many software development staffs.

The thin spread of domain knowledge in software projects is also observed in later years by Hofmann and Lehner (2001 [185]), and Damian et al. (2013 [109]). This has manifold implications for a software project.

- "... unexpected behavior might emerge in a system if a developer does not recognize any possible conflicting combinations between the system and the context."(Ubayashi et al., 2008 [372])
- Stakeholders may forget or be unaware of requirements, and analysts, without the deep DK of stakeholders, may not be able to fill in the gaps (Coughlan et al., 2003 [106]; Hadar, Soffer, and Kenzi, 2014 [165]). In consequence, "... requirement specifications are no longer complete due to the omission of relevant requirements for project success, and this could give rise to inconsistent specifications."(Pacheco and Garcia, 2012 [303])
- Missing contextual information leads to wrong decisions within the requirements engineering phase (Burnay, Jureta, and Faulkner, 2012 [90]; Azmeh et al., 2013 [28]), which impacts the whole development life cycle.
- Missing a structured way to elicit and document the context and related domain knowledge leads to communication overhead as domain knowledge has to be searched over and over again.(Damian et al., 2013 [109])
- Tasks are delayed when domain knowledge is not available.(Damian et al., 2013 [109])

Contrariwise, when conducting a structured context elicitation there are some significant benefits noted by researchers:

- Indeed the knowledge spread improves.(Curtis et al., 1988 [108])
- The understanding of a beforehand unknown domain improves significantly.(Garcia Alcazar and Monzon, 2000 [155])
- A unification of wording can be observed.(Garcia Alcazar and Monzon, 2000 [155])
- The usefulness and usability of the developed system-to-be improves. (Coughlan and Macredie, 2002 [105]; Azmeh et al., 2013 [28])

- "DK can compensate for a lack of clarity and missing information in the representation of a problem".(Coughlan and Macredie, 2002 [105]; Azmeh et al., 2013 [28])

When comparing the negative consequences and the benefits, we see that many benefits are the direct counterpart to a negative consequence (knowledge spread, understanding of the domain, unification of wording), while others give an additional surplus such as improved usability and usefulness. Hence, context elicitation indeed mitigates the risks missing domain knowledge bears and provides some surplus.

One could assume that the activity of context elicitation is well studied in research and also integrated into the best practices in industry as the importance of domain knowledge is known and acknowledged for a long time. Astonishingly, this assumption turns out to be wrong. Liu et al. (2010 [244]) studied the reasons for failing requirements engineering activities in China's software industry and state that one central reason for failure is that in general software engineers do not have access to sufficient domain knowledge and expertise. Additionally, also the requirements decision-makers lack of technical and domain expertise (Liu et al., 2010 [244]). Ubayashi et al. (2008 [372]), and Osada et al. (2007 [300]) also report that requirements analysis is mainly conducted from the viewpoint of systems functions and context and the related domain knowledge is only roughly analyzed. This general observation about the best practice in the industry is strengthened by a study as reported by Buchan and Hasana Ekadharmawan (2009 [78]), and Buchan et al. (2009 [79]) who analyzed the barriers to domain knowledge sharing in small and medium sized companies (SMEs). Buchan and Hasana Ekadharmawan (2009 [78]), and Buchan et al. (2009 [79]) state that for these kind of companies with very limited resources the elicitation and sharing of the context of a system-to-be is a big challenge which they cannot master right now. The main problems for SMEs, which work in the field of software engineering and development, are the selection of the right stakeholders to represent the clients' needs, a lack of a common vocabulary with the clients, missing sufficient representations of knowledge, and missing domain knowledge in general. Pacheco and Garcia (2012 [303]) analyzed the existing standards and guidelines which deal with requirements engineering and should give guidance for all important activities. For stakeholder identification as part of the context elicitation they observed

> Also, we can see that stakeholder identification in the requirements elicitation phase has received very little attention from the different existing initiatives in software development, for example in CMMi, SWEBOK, BABOK, and ISO/IEC 12207. All these initiatives recognize the existence of different types of stakeholders in the RE area. However, they only suggest examples and categories of stakeholders, and do not provide practices or guidelines to help and guide software engineers to identify stakeholders (who need to be identified in each project as an indispensable part of requirements elicitation).
>
> Despite its importance, the identification of stakeholders, including the identification of their needs and expectations, is poorly achieved in software projects.

We can summarize that context elicitation is not part of the current best practice in industry which causes many problems.

It could be argued that the best practice in industry follows with some delay the progress made in research. But in 2002, Offen (2002 [292]) states that

> So, given the ever increasing centrality of software as the system component that mediates the bulk of the functionality, performance, safety and reliability in so many contemporary systems, it is remarkable indeed that the roles and utility of this initial domain modelling process, alluded to above, are generally so poorly understood and manifestly underutilised by the software development community.

. Similar statements about the current state of research keep coming till today:

> This could not be further from the truth, however, as there is absolutely no agreement among experts on how best to elicit information or knowledge.(Davis et al., 2006 [112])

> Pure RE approaches have their origin in Software Engineering and they focus on the functionality the system must implement. This usually leads to the lack of specific processes for the elicitation of the human context and knowledge about what to gather.(Fuentes-Fernandez et al., 2010 [153])

Also Fabian, Gürses, Heisel, Santen, and Schmidt (2010 [131]) concluded in their survey about security requirements methods that it is not yet state of the art to consider domain knowledge. Consequently, socio-technical models capturing context are one of the milestones on the roadmap towards the unknown unknowns in requirements engineering defined by Sutcliffe and Sawyer (2013 [363]). We can conclude that the importance of context seems to be known, but currently this is not reflected in research for software engineering and requirements engineering methods in particular.

This observation seems to be paradox. Why is the importance of context elicitation acknowledged but not integrated into the state of the art in research and practice? One reason seems to be that domain knowledge describing the context is accepted as important input for requirements engineering and subsequent activities, but Pacheco and Garcia (2012 [303]), Sharp et al. (1999 [345]), and Ballejos and Montagna (2008 [31]) also observed that the elicitation of the context is often regarded as trivial task.

But there are many indications that the assumption that context elicitation is a trivial activity is wrong. In other engineering domains, a structured analysis of the context is common and an integrated part of the best practices (Offen, 2002 [292]). This might not directly imply that a elaborated context consideration has to be part of software engineering, because software engineering might be unlike other engineering fields. But the analysis of the current state of practice in the software industry shows that it actually is the case as missing domain knowledge is one of the big issues causing requirements engineering efforts to fail. And some further observations strengthens the impression that context elicitation is indeed far from trivial:

> The sharing of domain understanding is challenging in practice because of the inherent complexities. It may involve a large number of individuals with a broad diversity of existing specialized capabilities, expertise and vocabularies, as well as different cultures, beliefs and values.(Buchan and Hasana Ekadharmawan, 2009 [78])

> We learnt that users and customers expect engineering teams to know the domain very well. And if not, it was not easy for the teams to extract the characteristics of the domain, the restrictions, the constraints, and the exceptions.(Maalej and Ghaisas, 2014 [248])

> Software engineers can acquire domain knowledge by having discussions with customers. But customers occasionally neither mention any specifications of tacit assumptions nor any problems that might occur. Furthermore, domain knowledge becomes makeshift, cannot be reused, and becomes unstable when system developers have discussions to acquire domain knowledge without using any specific method. So we need technology that enables us to acquire domain knowledge and accumulate it.(Ugai and Aoyama, 2009 [373])

Such observations are seconded by the increasing number of stakeholders relevant for IT systems as these systems tend to become distributed spanning different organizations and groups

of customers.(Hentrich and Zdun, 2009 [182]; Pouloudi and Whitley, 1997 [316]; Buchan and Hasana Ekadharmawan, 2009 [78]) This is especially a challenge for SMEs as they do not have the resources to conduct a in depth analysis of the context (Buchan et al., 2009 [79]). All in all, we can say that the activity of context elicitation does not come by easily.

Up to this point we have argued that knowing the context and the domain knowledge which describes the context is important and brings in many benefits. But we do not want to conceal from the reader that there are also some reasons which indicate that having domain knowledge is not needed or even not beneficial in all situations.

Davis et al. (2006 [112]) observed that when using interviews to elicit requirements, the analyst who is conducting structured interviews does not necessarily has to be a domain expert. Hadar et al. (2014 [165]) even found some drawbacks of using domain experts as interviewers as they already have a fixed of point view biasing the course of the interview, some domain knowledge is not documented as both, the interviewer and the partner, perceive it as obvious, and even unwanted discussions within the interview in which the interviewer influences the point of view of his/her interview partner.

This observations for interviews also manifest as more general concerns regarding domain knowledge. Hadar et al. (2014 [165]) state regarding the overall requirements elicitation:

> However, another factor ... is that developers make assumptions about requirements instead of addressing questions to the users. Furthermore, users indicate that "developers know better" and create the information system they believe the users need, not that which the users requested. These two factors indicate a negative effect of a developer's DK on achieving a common understanding of the requirements.

Hadar et al. (2014 [165]) also observed that

> In our context, this could mean that analysts with high DK may overestimate their understanding of the requirements and thus underestimate the data that still need to be elicited, compared to their peers with lower DK.

Additionally, Berry (1995 [54]) states

> One of the potential benefits of this lack of knowledge is i.e. the abilities of a domain ignorant to state his or her ideas regardless of any domain assumptions and to ask revealing questions that could lead to unveiling issues that domain experts have overlooked.

This hypothesis was approved by a study in 2012 (Niknafs and Berry, 2012 [284]). Hence, involving domain experts as requirements engineers bears the risk of overlooked or not documented domain knowledge and biased requirements which do not reflect the users' needs.

Another point against experts with a high degree of domain knowledge is that they might not use the full solution space which leads to a fixation on certain solutions:

> Experts perform worse than novices when a shift from a standard means of representation is required or when a standard response is inappropriate.(Wiley, 1998 [390])

> Studies in the field of psychology show that DK may cause a tendency to approach situations in ways that have worked in the past and lead to fixation in problem-solving.(Hadar et al., 2014 [165])

Also, according to Wiley (1998 [390]), it seems that domain knowledge has a negative impact on creativity. Hence, requirements engineers with a high degree of domain knowledge for a problem at hand seem to focus on specific solutions and do not explore the full solution space in

a creative way. These risks might question the use of a context elicitation activity. But mixed teams turned out to be the most effective (Niknafs and Berry, 2012 [284]). Hence, at least one expert should be part of the team (Niknafs and Berry, 2012 [284]; Hadar et al., 2014 [165]). This indicates that the risks can be mitigated by just involving domain ignorant requirements engineers, which is in most cases a measure easy to take.

As a summary, we can state that context and the related domain knowledge is important, the benefits are significant, and the risks of having domain knowledge can be mitigated easily, but still there is a gap in research and current state of practice in requirements engineering.

6.3. The Idea of Patterns for Context Elicitation

As we have already noted, there is a gap regarding context elicitation in research and practice in requirements engineering. Hence, there is a need for means which support the elicitation of the context of a system-to-be and the related domain knowledge.

There are some challenges to master when designing a solution for context elicitation. The following can be found in literature:

Shared vocabulary A method should support the development of a shared vocabulary. Such a support should contain a basic vocabulary and a way to extend it.(Buchan and Hasana Ekadharmawan, 2009 [78]; Buchan et al., 2009 [79])

Abstract and detailed information The method should also provide support for externalizing and sharing of conceptually abstract as well as detailed information about the problem domain.(Buchan and Hasana Ekadharmawan, 2009 [78]; Buchan et al., 2009 [79]) An abstract representation is needed to get a quick overview and to enable the development of a basic understanding without a need for reading and comprehending details. But detailed information should not be lost as it is necessary for subsequent tasks. (Osada et al., 2007 [300])

Reconciling different view points The method should support the reconciliation of different view points from diverse stakeholders and accommodating changing and volatile understanding. Hence, the representation and documentation generated when applying a method should be easy to adjust and support the discussion between different stakeholders. (Buchan and Hasana Ekadharmawan, 2009 [78]; Buchan et al., 2009 [79])

Iterations Periodic verification of some representation of the domain shared between the stakeholders has to be supported.(Buchan and Hasana Ekadharmawan, 2009 [78]; Buchan et al., 2009 [79]) A complete view does not come by on the first iteration. Hence, the representation and documentation of domain knowledge should be easily adjustable, and iterations should be part of the method.

Models and Documents A method should combine graphical models as well as textual documentation for the documentation of domain knowledge (Curtis et al., 1988 [108]). While for a quick overview and for discussions a graphical model outperforms textual representations, details are easier to comprehend in a structured textual form (Osada et al., 2007 [300]).

Sufficient level of formalization Some degree of formalization might be helpful for analyzing purposes, but formalization always bears the risk of excluding certain stakeholders, because they might not be familiar with the formalism and they only have a small amount of time for learning something new (Garcia Alcazar and Monzon, 2000 [155]; Pinto-Albuquerque and Rashid, 2014 [313]; Karsai et al., 2014 [211]; Cheng and Atlee, 2007 [100]).

Lightweight notation The notation used should be a lightweight one (Garcia Alcazar and Monzon, 2000 [155]; Pinto-Albuquerque and Rashid, 2014 [313]; Karsai et al., 2014 [211]; Cheng and Atlee, 2007 [100]). It should

- only include the necessary domain concepts (Karsai et al., 2014 [211]),
- be consistent, and avoid unnecessary generality (Karsai et al., 2014 [211]),
- have a limited number of elements (Karsai et al., 2014 [211]),
- avoid conceptual redundancy (Karsai et al., 2014 [211]),
- make elements distinguishable (Karsai et al., 2014 [211]),
- use the same style everywhere (Karsai et al., 2014 [211]),
- enable modularity (Karsai et al., 2014 [211]),
- and adopt existing notations domain experts know (Karsai et al., 2014 [211]; Mohagheghi, Gilani, Stefanescu, Fernandez, Nordmoen, and Fritzsche, 2013 [269]; Cheng and Atlee, 2007 [100]).

Structured interviews Structured interviews and discussions gather more information than unstructured interviews.(Davis et al., 2006 [112]) Hence, structured interviews and discussions should be part of a method and supported by the means used within a method. A sufficient domain knowledge representation can help to set up such structured interviews.(Hadar et al., 2014 [165])

Avoid getting lost "There is a danger that too much time is spent on identifying roles and relationships, and the team is swamped with data. Knowing when to stop looking is just as important as knowing where to look."(Sharp et al., 1999 [345])

In a sense, we have gathered some requirements for a context elicitation method which we can use to assess a solution idea and an actual solution.

The central idea of our method is to make domain experts' knowledge available as an external input as involving experts is the benchmark regarding context elicitation:

> Experts generally solve problems in their fields more effectively than novices because their well-structured, easily activated knowledge allows for efficient search of a solution space. (Wiley, 1998 [390])

> An expert can typically recognize, store, and retrieve large meaningful chunks of domain-related information. An expert's processing is also commonly more abstract or conceptual than that of a novice. The proceduralization of an expert's knowledge base tends to allow for quick and easy access to memory and possible solution paths. Further, an expert's knowledge usually contributes to better problem representation, as experts are able to engage in a more qualitative analysis of a problem.(Wiley, 1998 [390])

> Through this interaction *[with documentation about a domain and discussions with experts]*, the analyst forms a mental model of the problem domain, from which requirements are developed.(Hadar et al., 2014 [165])

Patterns are one promising way to derive experts' knowledge and their way of thinking, and make it available to others:

> A technique that models the domain understanding at a conceptual level and is shareable, easily manipulated as well as (cognitively) accessible to both client and vendor stakeholders, could complement the more concrete representations such as a prototype. This would encourage deeper domain understanding and verification at a more conceptual level, and having the "big picture" front of mind when appropriate.(Buchan et al., 2009 [79]).

> The goal of pattern writing is to share expert's knowledge. In schema theory it is assumed that knowledge is stored in structural units, which are basically patterns in the mind of an individual. The pattern format therefore seems to be an adequate vehicle to capture these nuggets of wisdom. It is a form of representation that is very close to the way knowledge is clustered and discriminated in memory (according to schema theory). Patterns can, therefore, efficiently guide the reader to construct one's own schemas. It is important to note that, of course, a schema itself cannot be transferred. A pattern description only sketches the schema, but the understanding and capability of applying the pattern must be constructed by each individual. (Kohls and Scheiter, 2008 [225])

Hence, patterns can be used to transfer the knowledge an expert possesses to other people. Of course, this comes not by as easily as using, for example, a predefined question catalog which a requirements engineer can apply without any further effort, but making one building his/her own basic schema of a domain moves him/her as near to an expert as possible. There are additional reasons why patterns are a good choice in the context of context elicitation. Experts have problems to recall their knowledge explicitly when making decisions (Wiley, 1998 [390]), but patterns make such knowledge visible. Also the way how patterns are derived by looking at existing instances and merging them in an bottom-up way seems to be reasonable and even the "three occurrences make it a pattern" rule seems to have some foundation:

> ...the question arises how to build and maintain knowledge for advanced requirements elicitation systems. Our evaluation results provide evidence for the huge potential of following a bottom-up approach. (Meth et al., 2013 [268])

> Interestingly, we observed that the usage of retrieved knowledge outperformed the usage of imported knowledge already after three documents.(Meth et al., 2013 [268])

Of course, when using a pattern it is imported knowledge by itself, but as a pattern has to be instantiated it combines both, imported as well as retrieved knowledge. And patterns let one easily mix domain ignorance and domain knowledge as demanded by Niknafs and Berry (2012 [284]), as requirements engineers can be domain ignorant themselves and use the domain knowledge contained in a pattern to interact with the customer. Especially when resources are limited like in case of SMEs developing software and systems (Buchan et al., 2009 [79]), using patterns is favorable as the need for different domain experts is decreased. Additionally, the involvement of domain experts on both sides, the client side as well as on the software developers side, can be decreased as patterns give guidance, let one focus on the most important parts, and support a structured documentation. This is desirable as domain experts are a precious source, which tend to be overused when domain knowledge is not documented or the experts are interviewed over and over again as the needed knowledge is not clear (Damian et al., 2013 [109]). A pattern-based method also avoids the effect that "analysts with high DK may overestimate their understanding of the requirements and thus underestimate the data that still need to be elicited, compared to their peers with lower DK" (Hadar et al., 2014 [165]) as it makes explicit what has to be known and documented for a pattern instance. For requirement engineers without domain knowledge,

patterns turn the unknown unknowns (Sutcliffe and Sawyer, 2013 [363]) into known unknowns which is desirable as Sutcliffe and Sawyer (2013 [363]) state:

> In this case the analyst has some awareness of the necessary knowledge, so an agenda for elicitation can be set. Most techniques involve exploring the implications of the system-domain boundary, exemplified by Jackson's formulation of the RE problem.

As a result, we can state that the idea of using patterns for context elicitation looks promising.

There is only one concern regarding methods which rely on abstraction as formulated by Hutchinson et al. (2011 [190]):

> ..., it is by no means guaranteed that higher abstraction levels lead to better software. In fact, results from psychology generally and psychology of programming specifically show that abstraction can have a negative effect because thinking in abstract terms is hard, with a tendency for individuals to prefer concrete instantiations (e.g., exemplars, simulations) over abstract conceptualizations.

This risk also applies for patterns. Hence, there is a need to explain the elements of a pattern and also a need for a method which shows how to use the pattern. Examples should be used to ground the abstract concept in something the user is familiar with. When keeping this additional requirements in mind, patterns seem a way to go for context elicitation.

But can a pattern-based method in general fulfill our requirements for a context elicitation method?

Shared vocabulary When building a pattern also a basic vocabulary is introduced as we will see in Chapter 7[16]. But the expressiveness of the vocabulary depends on the actual patterns. Hence, we get a yes, but keep in mind that the vocabulary might be limited.

Abstract and detailed information On the level of a pattern description this requirement is fulfilled. A pattern description contains an abstract overview as well as the details for understanding and applying it. If a pattern instance contains abstract as well as detailed information, is dependent on the pattern itself. We get a yes for the pattern description, but for each pattern we have to check whether an instance also fulfills the requirement.

Reconciling different view points This purely depends on the actual patterns.

Iterations This purely depends on the methods which come along with the actual patterns.

Models and Documents It is common for patterns to combine graphical representations and textual parts for describing patterns as well as for describing instances. Hence, here we get a yes as long as we stick to the common practice in the patterns community.

Sufficient level of formalization Patterns are for most parts very informal. Only the graphical representation is often presented using a semi-formal notation. This is regarded as sufficient for patterns in general. To which degree this is sufficient for our purpose has to be judged when looking at the actual patterns.

Lightweight notation The notations used in pattern descriptions are lightweight and are crafted for the target audience. Pattern notations

- only include the necessary domain concepts as these are the elements which describe a pattern,

[16] Page 113

- avoid unnecessary generality as a pattern wants to be as concrete as possible without loosing the structure which makes it a pattern,
- have a limited number of elements as it focuses on the core elements which form the pattern,
- avoid conceptual redundancy as elements with the same concept are merged,
- and enable modularity as patterns are organized in so called pattern languages which allow the combination of patterns.

Hence, we can conclude that patterns have the potential to fulfill this requirement as long as we stick to the best practice. Nevertheless, the actual patterns have to be defined and applied appropriately.

Structured interviews Patterns structure the important information and can be used for setting up structured interviews.

Avoid getting lost Patterns narrow down a solution to the important elements and their relations. Hence, they provide guidance and avoid getting lost.

From this discussion, we see that patterns are in general suitable to fulfill our requirements for a context elicitation method, but we have to check again with our actual patterns at hand.

6.4. Related Work

As already discussed in Chapter 4[17], we use mapping studies to identify gaps in research and position our work in the scientific body of knowledge. Section 6.2[18] and Section 6.3 are based on the insights gained from a mapping study in the field of context elicitation for requirements engineering. How we conducted the mapping study is explained in detail in Section 4.2.4[19]. In Section 6.4.1 we present some more information gained from the mapping study. In Section 6.4.2 we review the related work in the field of context elicitation for requirements engineering, while in Section 6.4.3[20] we review the related work for mining and writing patterns and pattern languages.

6.4.1. Results of a Problem & Gap Study about Context Elicitation in Requirements Engineering

Some further insights, which were not already discussed in the preceding sections, from the analysis of the literature found for a problem & gap study in context elicitation for requirements engineering are presented in this section. The problem & gap study was conducted using a manual search in the publications of selected conferences and workshops spanning the years of 2009 until today[21]. The initially found literature was extended by snowballing[22] without any limit. For more details see Section 4[23].

Some numbers regarding the search are shown in Figure 6.2. After scanning the titles and abstracts of more than 8000 papers found while searching the selected venues, 68 papers were selected as potentially related to context elicitation in requirements engineering. Additionally, 58

[17] Page 67
[18] Page 82
[19] Page 70
[20] Page 100
[21] The last iteration was done in March 2015
[22] The activity of analyzing the bibliography of an paper already identified as relevant to find further candidates is called snowballing.
[23] Page 67

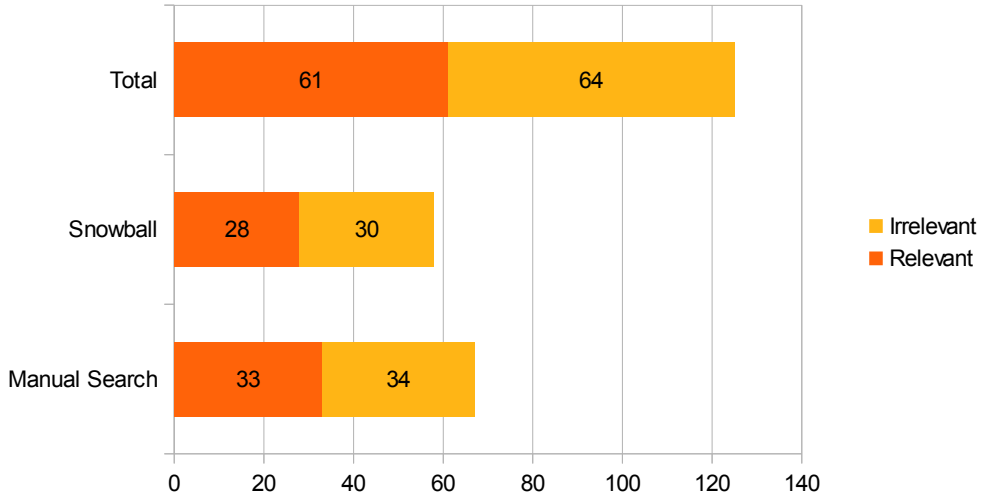


Figure 6.2.: *Found Literature*

were included after analyzing the bibliography of already included papers (snowballing). After reading the full papers, 33 papers of the manual search and 28 found while snowballing were selected as relevant papers. Hence, we found 61 relevant papers in total[24].

Of course, these papers discuss the importance of context elicitation in requirements engineering or state a specific problem related to context elicitation. A statement about the importance of context elicitation describes that and why context elicitation has to be considered without stating an actual problem connected to this topic. In contrast, a problem statement describes a

[24]The list of relevant papers and the information collected for them is shown in the Appendix A.3.2[25]

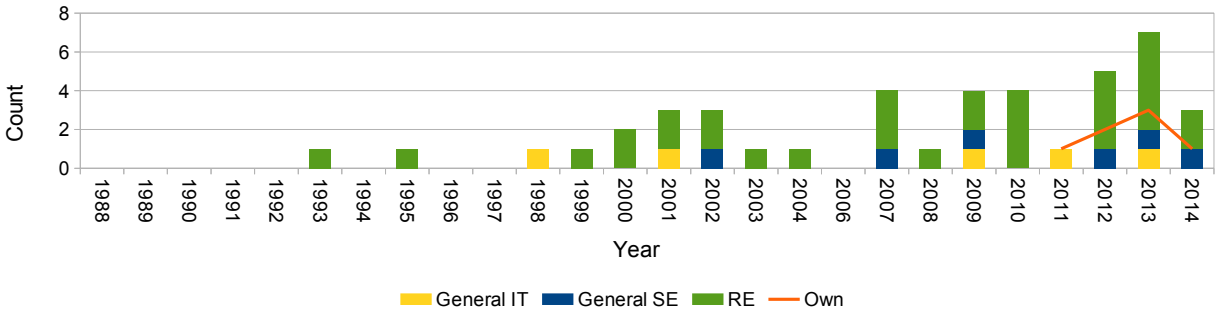


Figure 6.3.: *Distribution of Statements Highlighting the Importance of Context Elicitation over Time*

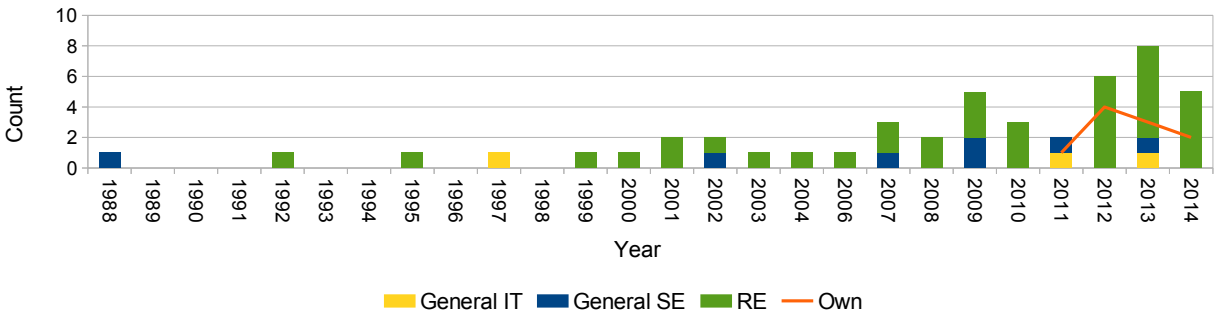


Figure 6.4.: *Distribution of Problem Statements Regarding Context Elicitation over Time*

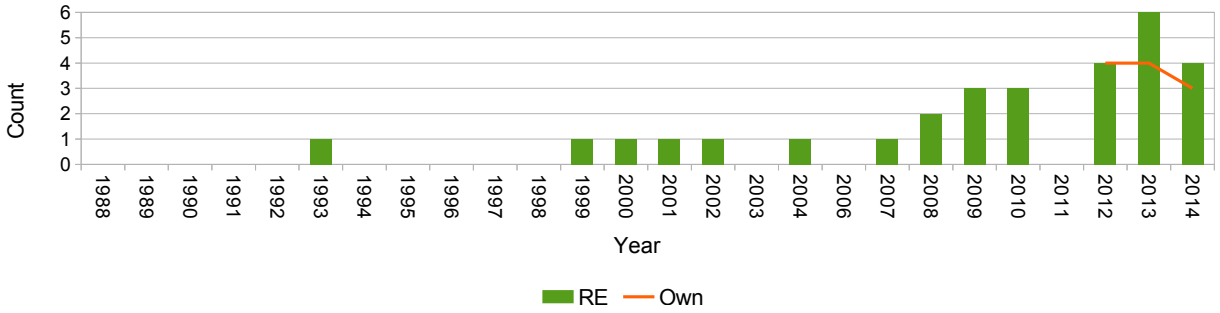


Figure 6.5.: *Distribution of Solution for Context Elicitation over Time*

concrete issue related to context elicitation, but does not necessarily highlight the importance of context elicitation as such. Figure 6.3 and Figure 6.4 show the distribution of such statements over time. A yellow bar expresses that a statement was given in general for IT systems, a blue bar that it was stated in general for software engineering, and a green bar that it was stated specifically for requirements engineering. The orange line indicates the number statements which were coined in one of our papers. These works are also included in the bars and we want the reader to be aware of the fact that some trends visible in the charts are influenced by our works. What we see from the numbers is that the topic was known since the late 1980s, and got some additional attention around 2000. In the recent years the topic of context elicitation became a hot topic. Note that the difference between the recent years and the years before might be a bit flawed as the manual search in related venues was only done for the years 2009 until now. Hence, it might be that snowballing only reveals the most significant works in the years before, while a manual search might have revealed some additional papers for these years. But we already see an positive trend starting in 2007. Hence, from our point of view it is valid to say that context elicitation got increasing attention lately and that not only the importance is highlighted but also significant problems related to context elicitation are still existing.

It is not surprising that the problems related to context elicitation are not resolved even after more than 20 years since context elicitation was discovered as important topic for the first time when looking at the actual solutions proposed in science during this time span. Figure 6.5 shows the proposed solutions and their distribution over time. Note that we count each publication as a solution. We did not analyze whether several publications form only one solution. Until 2008, only 7 solutions were proposed which directly considered context elicitation. Starting from 2008, some more solutions were proposed and here the difference to the years before is obviously

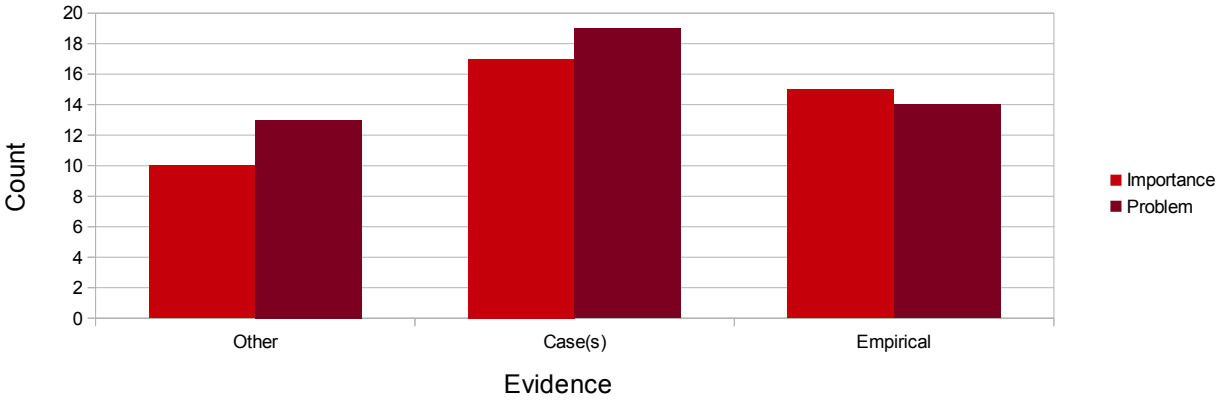


Figure 6.6.: *Evidence of Statements about Importance and Problems*

significant, even when considering the differences between manual search and snowballing. But we can also notice a slight decrease of solutions published in the latest years, when removing our publications on the topic.

One might now ask which evidence is given by the papers for the statements about the importance and problems of context elicitation in requirements engineering. Figure 6.6 shows which evidence the papers provide for their statements. We distinguish between papers which provide results from some empirical work on which they ground their statements, papers which investigated the matter of context elicitation in real cases (which were freely available or provided by industrial partners), and papers which ground their statements on own experiences, toy examples and the like. What we can see from Figure 6.6 is that the mayor part of papers are based at least on a real case and still a significant part is based on empirical work. We can conclude that we have a strong evidence for the importance of context elicitation and the existing problems.

Looking at the solutions and which phase of requirements engineering are covered we get a surprising result (see Figure 6.7). We distinguish three general phases: the initial phase in which the sources of information which are relevant are identified and first information is gathered, the early phase which includes high level activities such as goal modeling, and the late phase

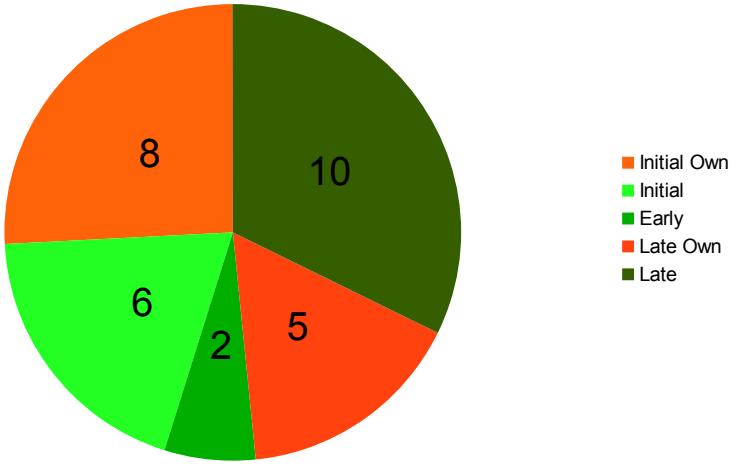


Figure 6.7.: *Requirements Engineering Phases Covered by Solutions*

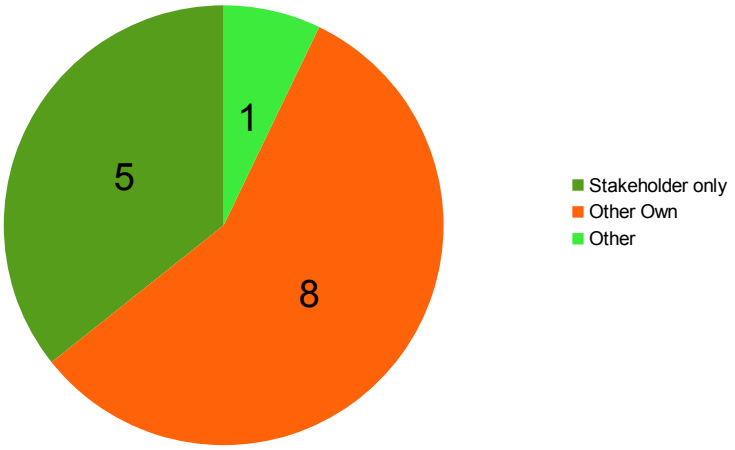


Figure 6.8.: *Focus of Solutions in the Initial Phase*

in which the actual system requirements are documented, analyzed, and refined. Additionally, we distinguish between papers written by us (given in shades of orange) and papers provided by others (given in shades of green). Most statements about context elicitation highlight that context elicitation and the related domain knowledge documentation should be one of the first activities to be taken as it sets the boundaries of the problem space. Leaving our proposed solutions aside, the coverage of phases is just the other way round. More than half of the solutions cover the late phase. Hence, without our works, the initial phase is underrepresented compared to its importance.

Taking a closer look at solutions in the initial phase (see Figure 6.8, we see that there is a strong focus on stakeholders. Five out of six solutions which are not proposed by us focus on the stakeholders only. Hence, stakeholder elicitation seems to be a part of context elicitation on its own. Still, the problem statements do not only highlight the need for stakeholder elicitation, but the complete context including stakeholders. Again, without our works, the importance of the complete context is not reflected by the solutions available.

A similar imbalance can be observed for the covered activities related to context elicitation. Figure 6.9 shows that most solutions, which are not proposed by us, focus on the modeling and documentation, and/or refinement of already discovered information about the context[26]. Only six of the solutions proposed by other researches actually identify the context in the first place and do not require the context to be already known.

The next matter we analyzed through our problem & gap study are the parts of the context

[26]Note that a solution can cover several activities, views, and even phases at a time

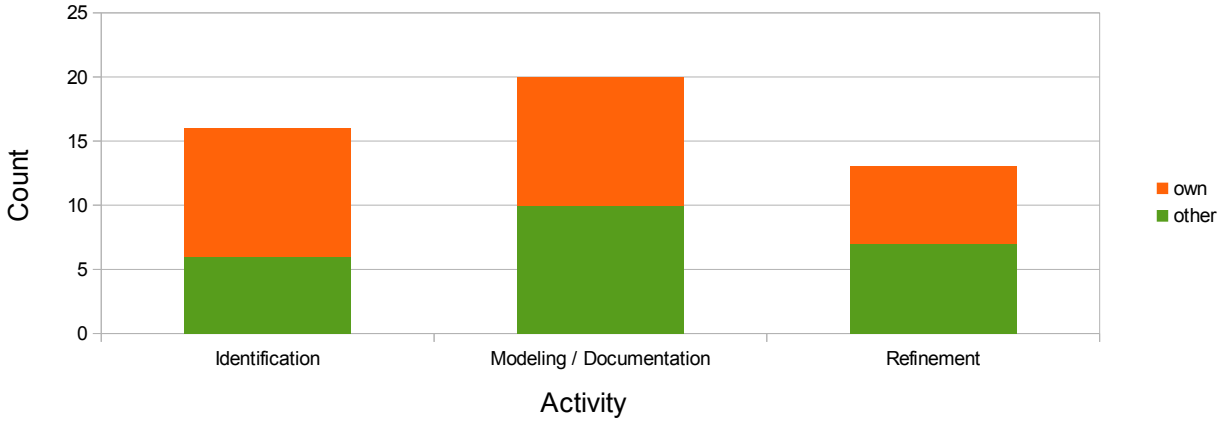


Figure 6.9.: *Activities Covered by Solutions in the Field of Context Elicitation*

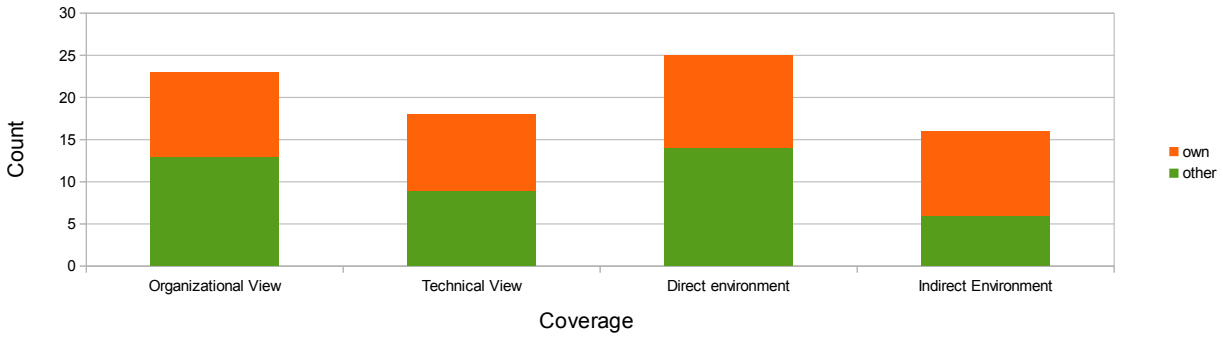


Figure 6.10.: *Areas of Context and the related Domain Knowledge Covered by the Solutions*

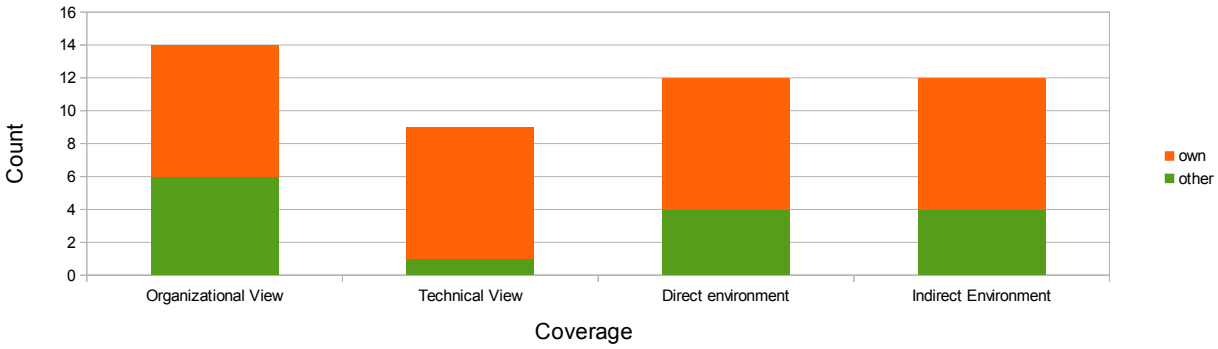


Figure 6.11.: *Areas of Context and the related Domain Knowledge Covered by the Solutions for the Initial Phase*

which are covered by solutions. Here, we distinguish between the technical view including, for example, already operating systems, technical infrastructure and alike, the organizational view including, for example, processes and stakeholders, the direct environment which includes all entities directly related to the system-to-be, and the indirect environment which includes entities which only implicitly have relations to the system-to-be. Figure 6.10 shows the result of this analysis. We see that the organizational view is as well covered as the technical view. But there is a noticeable focus on the direct environment. Looking at the initial phase only (Figure 6.11), we again observe that the technical view is neglected as only one solution proposed by other authors provides this view.

Last, we investigated which of the solutions provided are domain specific. Beside our own, only one further solution had a domain-specific view.

From the results of our problem & gap study we can conclude that our solutions for the elicitation of context in requirements engineering fills some gaps. First of all, our context patterns cover the initial phase of requirements engineering allowing the identification of the important elements of the context covering the technical as well as the organizational view, are domain-specific, and the patterns do not only focus on the direct environment but also take the indirect environment into account.

6.4.2. Context Elicitation

In this section we discuss the related work which describes solutions regarding context elicitation.

Initial Phase: Stakeholder Identification. Ballejos and Montagna (2008 [31]) present a method to identify stakeholders which are of importance for the requirements engineering of a system-to-be. The method starts with a specification of stakeholder types. These types are derived for each system-to-be individually, and the method does not provide detailed guidance, beside a basic template which can be used to characterize a stakeholder type, for this step. Then stakeholder roles have to be specified. Ballejos and Montagna (2008 [31]) provide a catalog of predefined roles which can be used to derive the stakeholder roles for the actual system-to-be by using a stakeholder role template. Afterward, stakeholder roles are assigned to the identified stakeholder types. Last, the relations between the stakeholders are elicited and analyzed. Sadiq and Jain (2014 [331]) adopt the process of Ballejos and Montagna (2008 [31]). They use then fuzzy logic to formalize the stakeholders found, and to group and prioritize them. The methods as described by Ballejos and Montagna (2008 [31]), and Sadiq and Jain (2014 [331]) require some effort to be spent on the first step when starting without prior knowledge. Additionally, the authors state that the found stakeholder types identified were increasing significantly with the size of the project, and that they are in need for a way to abstract and cluster stakeholder types.

Hence, for both problems a context pattern might help as it provides already initial knowledge and abstract stakeholder types. In turn, the works of Ballejos and Montagna (2008 [31]), and Sadiq and Jain (2014 [331]) can be used along with context patterns whenever a more fine grained view on stakeholders is needed. Another work concerned with stakeholder identification is the one of Alexander and Robertson (2004 [13]). They highlight the importance of visualizing and characterizing stakeholders to understand their impact on requirements engineering within a software development project. They propose a so called onion model to visualize and characterize stakeholders. The onion has the system-to-be in the center. It is surrounded by a layer containing stakeholders interacting directly with the system. The next layer includes the stakeholders which use other systems which have an interface with the system-to-be. The last layer includes those stakeholders which only influence already identified stakeholders. Alexander and Robertson (2004 [13]) have some suggestions how to find the actual stakeholders, but focus on the visualization itself. One of the suggestions is to use domain-specific prepopulated onion models. In a way, our context patterns share the idea of different environments for a system-to-be and assigning stakeholders to these environment. In this sense, context patterns follow the basic idea of onion models. Additionally, they do not only focus on stakeholders and they are domain-specific and prepopulated. Sharp et al. (1999 [345]) is also focused on identifying stakeholders within any requirements engineering process. Sharp et al. (1999 [345]) provide four kinds of so called baseline stakeholders, which are then refined to the actual stakeholders of a system-to-be in a five-step process. The baseline stakeholders are users, developers, legislators, and decision-makers. The method proposed by Sharp et al. (1999 [345]) is rather generic and can be applied to any domain. But in turn, the guidance provided for the five steps is minimalistic describing what to do but not how to do it in detail. The method of Sharp et al. (1999 [345]) is an option whenever no context pattern is available, the effort for deriving a new context pattern cannot be spent, and only the stakeholders are of interest.

Early Phase: Goals. Singh and Woo (2009 [355]) propose five guiding principles which can be used to move from a strategic business goal and IT landscape level to goal models used in requirements engineering. In this way, they want to assure the alignment between the system-to-be and the strategic business goals. For the business level everything has already to be documented as the principles only describe the transition from the initial documents to the goals for a system-to-be. The principles are rather high level describing only what to do, but not how to do it such as "identify high-order constructs", "identification of assigned goals", and so forth. Singh and Woo (2009 [355]) proposal covers direct and indirect environments as long as they are part of the initial documents, but the proposal focuses on the stakeholders and their goals, neglecting the technical perspective. Our context patterns can be used when following the proposal of Singh and Woo (2009 [355]) because a context pattern instance can be one of the necessary inputs, as it contains some of the initial information needed, such as business stakeholders and high-order goals.

Early Phase: Refining the Social Context. Fuentes-Fernandez et al. (2010 [153]) focus on the social context of a system-to-be. They aim at extending existing context descriptions with information about social relations and social implications. They provide a catalog of areas, aspects of importance within an area, and social properties bound to an aspect. These catalogs are used to identify the relevant social properties for a system-to-be. For each social property patterns as well as anti-patterns exist to elicit important information about the property at hand for the actual system-to-be. The patterns not only cover the context information to be elicited but also the implications for, later on, elicited requirements. A similar method is outlined by Proynova et al. (2010 [317]), which focuses on the personal values of each stakeholder. These values have to be discovered as they can influence the perception of requirements. The method itself can hardly be judged as only a research preview exists by now. The works of Fuentes-Fernandez et al. (2010 [153]), and Proynova et al. (2010 [317]) are complementary to our context

patterns, as on the one hand the social relations are investigated more fine grained than in our context patterns, but on the other hand the work can only be applied if the general context is already established, for which our context patterns are sufficient.

Late Phase: Stakeholder Identification. Castro-Herrera and Cleland-Huang (2009 [97]) present a machine learning approach which enables the identification of stakeholders of a system-to-be. The according tool is capable of retrieving direct as well as indirect stakeholders from existing documentation for a system-to-be. It identifies relevant topics and the stakeholders which are interested in these topics or contributed somehow to a topic. It can also score the level of expertise for a stakeholder and a topic and generates a graph of topics and how they influence each other. For the tool support, one needs the requirements as well as the persons who contributed the requirements as input. Azmeh et al. (2013 [28]) also present an approach to identify stakeholders and groups of stakeholders which are of importance for certain requirements. The approach is also based on already documented requirements and a documentation which reveals the users involved in writing a requirement. The users are then grouped according to the general stakeholder they represent and also the general stakeholders are related to each other. The approaches of Castro-Herrera and Cleland-Huang (2009 [97]), and Azmeh et al. (2013 [28]) can be used along with ours. They do not help to elicit the initial context and requirements, but they can be used to discover stakeholders, which might have been overlooked in the beginning. Additionally, in some cases it might even be useful for the analysis of documents when deriving a new context pattern.

Late Phase: Modeling and Refining the Direct Environment.
Jackson and Zave (1993 [201]) propose to model a system-to-be in terms of its context and the domains it contains, and the problems the system-to-be shall solve. They also introduce reoccurring problem patterns, the problem frames. This early work proposes to use whatever modeling and specification notation serves the current purpose best. In (Jackson and Zave, 1993 [201]) they use the Z notation. Later on, this initial proposal was refined to the problem frames notation and method (Jackson, 2001 [200]), which is described in detail in Section 2.2.4[27]. Ubayashi et al. (2008 [372]) present a UML profile to model embedded systems and their context. The profile focuses on the actuators and sensors which are part of the system-to-be and the context which is influenced and measured respectively. The interaction with the context can be characterized by annotating certain types to the associations between the elements of the context and the actuators and sensors of the system-to-be. The notation has a purely technical focus. Ubayashi et al. (2008 [372]) themselves state that their notation is somewhat comparable to the problem frames notation (Jackson, 2001 [200]), but their notation is focused on embedded systems and technical elements only and they do not provide any problem patterns. Unlike for the original problem frames notation, Ubayashi et al. (2008 [372]) provide tool support for model checking and deriving test cases. Ubayashi et al. (2008 [372]), Jackson and Zave (1993 [201]), and Jackson (2001 [200]) focus on the direct environment only whereas our proposal has a larger focus. Moreover, Ubayashi et al. (2008 [372]), Jackson and Zave (1993 [201]), and Jackson (2001 [200]) only provide the means to model the context on a detailed level for already existing requirements, but do not provide any guidance how to actually discover the context and requirements, their methods can be seen as complementary to our proposal.

Late Phase: Using And Refining Context in Requirements Discussions. Leonardi et al. (2010 [236]) present a phenomenon called "Ahab's Leg" which occurs when using scenario-based techniques for discussing and validating requirements in collaboration with the according stakeholders. Basically, context descriptions which are only introduced and used to make a scenario, which is crafted to describe a certain requirement or a group of requirements, more lively, can distract the stakeholders, leading to unwanted discussions. This phenomenon is called "Ahab's leg". The authors provide some insights into this phenomenon and derive some rules

[27] Page 35

how to deal with such misleading context information. In this sense, Leonardi et al. (2010 [236]) talk only about refining the context carefully when crafting scenarios for already existing requirements, but they are not concerned with the initial context elicitation like our proposal is.

Late Phase: Refining and Improving Natural Language Requirements based on Context. Annervaz et al. (2013 [17]) present a method to improve the analysis of natural language requirements using domain models which describe the context of a system-to-be. The method itself is agnostic regarding the notation of the initial domain model. The domain model is used to build an ontology of important terms and their relations to each other. The important actual terms used for describing a system-to-be have to mapped to this ontology. This information is then used to increase the precision and recall of natural language processing (NLP) of requirement texts. Again, the requirements and the context has already to be known for applying the method. The method of Annervaz et al. (2013 [17]) only refines and structures the context in a way it is suitable for NLP. Our context patterns provide not only the domain model itself, but also the mapping between the actual terms and the model, as a context pattern instance is expressed in terms of the system-to-be at hand.

Late Phase: Integrating Context Elicitation for Requirements Engineering with complete SE Life-Cycle. Garcia Alcazar and Monzon (2000 [155]) present an experience report about a requirements analysis and specification framework they use within their company. This framework describes several steps, such as domain analysis or requirements documentation, which are of importance for successful requirements engineering. The framework itself does not define how to execute the steps themselves or the sequence of the steps, but only their purpose, and inputs and outputs. One cornerstone of their framework is to consider the context of the system-to-be thoroughly. As the framework neither gives any details nor proposes any method for conducting a single step, our method can be chosen for the steps which include context elicitation. Ben Mena et al. (2007 [52]) present a way to integrate context awareness as crosscutting aspect into a complete software development life-cycle. Therefore, they identify the touch-points between the general software engineering activities and the context of the system-to-be and outline which models are necessary for which activity. Of course, the development life-cycle includes also requirements engineering, but as matter of fact, Ben Mena et al. (2007 [52]) derive all context information from the results of the requirements engineering phase and give no guidance on how to elicit this information. Hence, the method is for refining and restructuring context information gained while conducting requirements engineering to be suitable for downstream activities. Therefore, our context patterns can be used as input to the method of Ben Mena et al. (2007 [52]).

Late Phase: Documenting gathered Domain Knowledge.
Ugai and Aoyama (2009 [373]) propose a special kind of wiki for documenting domain knowledge. The basic form they use to describe domain knowledge is based on the insights gained from analyzing pattern forms. The authors propose to use such a wiki in a closed community such as a company. They highlight the importance of rules and incentives to use such a wiki because the initial motivation for documenting domain knowledge is rather low. The wiki itself targets general documenting purposes and is not directly focused on supporting requirements engineering for a concrete system-to-be. But the existence of such a wiki could be useful as a source for new context patterns as well as a source for instantiating a context pattern for a system-to-be. Unfortunately, no such wiki exists which is publicly available.

6.4.3. Patterns

In this section we discuss related work on the topics of pattern writing and pattern languages.

Deriving Patterns Baumgartner and Kohls (2013 [33]) discuss in their paper the general

problem of finding the right level of abstraction for a derived pattern. They first discuss the necessity of abstraction when deriving a pattern, and give some illustrating examples for descriptions being too abstract or too detailed. Then they discuss different levels of abstraction, and how to characterize a level of abstraction. This is followed by a definition of different abstraction strategies. They conclude with suggestions for selecting the right level of abstraction. As the work of Baumgartner and Kohls (2013 [33]) was recently published, our work on context patterns was not directly inspired by the insights provided by Baumgartner and Kohls (2013 [33]). But we checked whether the knowledge provided by Baumgartner and Kohls (2013 [33]) would have changed our course of action. At some minor points, there might be improvements possible which are highlighted in the according text, but for most points the insights from Baumgartner and Kohls (2013 [33]) just strengthened our own method. But nevertheless, Baumgartner and Kohls (2013 [33]) provide interesting insights for readers when starting with deriving patterns on their own. Rising (1998 [321]) describes some best practices for mining patterns. We integrated and applied them all within our method for deriving new context patterns. We mined our own experiences, which includes making new experiences, we mined available documents (mining by borrowing), and we used meetings and workshops for mining.

Writing Patterns Meszaros and Doble (1997 [267]) provide a pattern language on pattern writing. The contained patterns highlight some common problems when writing a pattern and solutions for these problems. The patterns are very abstract. Hence, they do not replace reading patterns by other authors to get a basic understanding of what makes a pattern. Nevertheless, the patterns are helpful as basis to assess if everything important was considered while writing a pattern. Kohls (2012 [224]) uses the path metaphor to describe how to carve out the actual pattern from some found instances. For this matter, he describes a sequence of steps to follow. The steps are: understanding the context, understanding the problem and forces, understanding the solution, understanding the consequences, and understanding the abstraction. In a second work, Kohls (2011 [223]) describes some basic qualities a pattern description should have and from which characteristics for a quality one can choose from. These works influenced us when writing our patterns and is recommended for readers who want to write an own context pattern.

Models in Patterns Schäfer et al. (2011 [338]) describe patterns for creating graphical domain specific languages (DSL). The patterns themselves are rather high-level describing, for example, how to represent aggregations. As DSLs share some commonalities with our context patterns (see Section 7.3[28]) at least for the graphical part, these patterns were handy to assess our graphical representation under the light of alternatives for, for example, representing aggregations. The work of Karsai et al. (2014 [211]) is also from the field of domain specific languages, but a couple of the guidelines they propose for writing domain specific languages are also applicable to our context patterns. We already considered those guidelines as requirements for our context patterns as discussed in Section 6.3[29]. From the field of meta modeling is the work of Cho and Gray (2011 [101]), which defines some basic design patterns for meta-models. They are rather high level and do not uncover any astonishing new modeling options, for example, that a relation can be represented as attribute or as intermediate class. But we used them to assess our own meta-model for context patterns to make sure that we used the most appropriate way of modeling.

Relating Pattern Winn and Calder (2003 [392]) describe a pattern language for pattern languages in their work. It contains different patterns which can be applied to solve different problems occurring while deriving a pattern language. While some of the patterns emphasize the importance of relations between patterns, no pattern is available for the problem of finding the relations. Our method for relating patterns which we will describe in Section 9.1.4[30] reflects some

[28] Page 120
[29] Page 87
[30] Page 166

of the patterns described by Winn and Calder (2003 [392]), but is much more detailed, specifically tailored to context patterns, and explains how to find relations. Pauwels et al. (2010 [307]) also stress the importance of relations between patterns. But the method for building a pattern language contains no explicit step for finding relations, nor does any other step embody the relation mining. Hence, it is somehow related to our work on context patterns and a context pattern language, but does not cover all aspects we will elaborate in the following chapters. Zdun (2007 [400]) proposes, based on an idea of Henney (2005 [181]), to use formal grammars to refine and define pattern relations and subsequently to use the relations for pattern selection. While existing relations are refined when formalizing them and additional information for the relations is collected, the initial set of relations has to be known beforehand. Hence, this work might be fruitful for further investigations and refinement of relations between context patterns, but the authors state themselves that the usability and benefits in practice turned out to be very limited. Hafiz et al. (2012 [166]) present a pattern language for security patterns and also give some insights how they built the pattern language. Their method description and lessons learned are more of anecdotal quality than being a detailed process one can follow. Hence, it inspired parts of our work but is not explicitly reflected in our method of relating patterns.

Domain Specific Patterns Endrei et al. (2004 [127]) present SOA related patterns derived from their work in SOA projects at IBM. The presented patterns include a high-level business view as well as a detailed technical view. The patterns are described from an IT landscape point of view, which means they focus on the technologies and parts which are needed to run a SOA rather than deep diving into building a specific service. The works of IBM on the matter of SOA which are freely available were one main input for deriving our SOA context patterns. Hentrich and Zdun (2009 [182]) present a pattern language for the integration of processes into the design of a SOA. These patterns provide a technical view on the problem and focus on supporting design and implementation decisions. Nevertheless, business processes and involved actors are part of the context of a system-to-be. Hence, some patterns touch the topic of context but they do not focus on its elicitation and alike. But as the patterns provide general information about technical parts of a SOA and some stakeholders, we used this work as one of our inputs for our SOA related context patterns. Lytra et al. (2012 [246]) also provide some useful insights we used for deriving our context patterns. The pattern language they present is on the topic of integrating different platforms using service-based technologies. Of course there are some more works on SOA patterns (for example, Khan, Kästner, Köppen, and Saake (2011 [212])), which focus on the technical details of a SOA and which provide solutions for design problems or even issues with code, but we did not consider them in detail as they were too low level for being useful or the information provided was already covered by the considered works. Our literature review did not reveal any pattern-based methods for the topics of smart grid and law consideration. The general related work to legal requirements engineering is discussed in Section 14.3[31].

6.5. Towards a Pattern Language for Context Elicitation

As already discussed in Section 6.3[32], we propose to use patterns for context elicitation. These patterns are domain-specific. Hence, one context pattern in isolation only helps in eliciting the context related to this particular domain. But the context of a system-to-be is often not only related to one domain, but several domains. Hence, a combination of context patterns is necessary. A pattern language helps in finding and combining the necessary pattern for describing the complete relevant context of a system-to-be. In this sense, one pattern is only a piece in the overall puzzle which describes the context. In the following, we will elaborate

[31] Page 242
[32] Page 87

what constitutes a pattern language, and how a pattern should be described to fit in the overall context puzzle.

Alexander (1977 [12]) had the initial idea to describe common design solutions in patterns. Section 6.5.1 describes Alexander's definition of a pattern language. From these insights we derive in Section 6.5.2 an own template stating what has to be described to form a pattern language. In fact, Alexander was an architect, who designed buildings and in Sect. 6.5.3 we illustrate how software engineers define a pattern language. We relate these definitions to the definition of Alexander.

6.5.1. Alexander's Definition of a Pattern Language

Alexander (1977 [12]) described the term *pattern language*, which is a structured method for describing common design practices for a knowledge area. Alexander described a pattern language for creating towns and buildings in Alexander (1977 [12]) and he wanted to empower ordinary people to successfully solve very large, complex design problems. "This language is extremely practical. It is a language that we have distilled from our own building and planning efforts over the last eight years. You can use it to work with your neighbors, to improve your town and neighborhood. You can use it to design a house yourself, with your family; or to work with other people to design an office or a workshop or a public building like a school. And you can use it to guide you in the actual process of construction." (Alexander, 1977 [12], p. x).

Inspired by the work of Alexander we looked into the essential elements of a pattern language and state that these elements are vocabulary, syntax, and grammar. Note that Alexander did not explicitly state in his work that these are elements of a pattern language, but we argue in the following that these elements are referenced in his work.

Beforehand, we define these terms for human language based on the Oxford English Dictionary (OED). The term language in the OED[33] is defined as follows: "The system of spoken or written communication used by a particular country, people, community, etc., typically consisting of words used within a regular grammatical and syntactic structure". In addition, the OED[34] defines the vocabulary of a language as: "the body of words used in a particular language". Moreover, the OED[35] defines the term semantic as: "relating to meaning in language or logic". Last, the OED[36] defines the term grammar as "Grammar is the way in which words are put together to form proper sentences." and the term syntax[37] "The arrangement of words and phrases to create well-formed sentences in a language". *Hence, the grammar is used to produce a syntactically correct sentence from a set a words which have a specific semantic.*

Alexander states in regard to a pattern language that "the elements of this language are entities called patterns." (Alexander, 1977 [12], p. x). Hence, patterns are the words forming the vocabulary of a pattern language. In addition, Alexander states that "A pattern language has the structure of a network. [...] However, when we use the network of a language, we always use it as a sequence, going through the patterns, moving always from the larger patterns to the smaller, always from the ones which create structures, to the ones which then embellish those structures, and then to those which embellish the embellishments. ... Since the language is in truth a network, there is no one sequence which perfectly captures it. But the sequence

[33] The definition of the term language in the Oxford English Dictionary: `http://www.oed.com/view/Entry/105582?rskey=uuYVrM&result=1&isAdvanced=false`

[34] The term vocabulary defined in the Oxford dictionaries `http://www.oxforddictionaries.com/definition/english/vocabulary?q=vocabulary`

[35] The definition of the term semantic in the Oxford dictionaries `http://www.oxforddictionaries.com/definition/english/semantic`

[36] The definition of the term grammar in the Oxford dictionaries `http://www.oxforddictionaries.com/words/grammar`

[37] The definition of the term syntax in the Oxford dictionaries `http://www.oxforddictionaries.com/words/syntax`

which follows, captures the broad sweep of the full network; in doing so, it follows a line, dips down, dips up again, and follows an irregular course, a little like a needle following a tapestry." (Alexander, 1977 [12], p. xviii). Hence, a pattern language also has syntax given by the direct relations between pattern and a grammar defined as valid sequences. *A well-formed solution is then formed by using the sequences of patterns (grammar), which are correct with regards to the relations between the patterns (syntax), to combine patterns (words) in a meaningful way (semantic).*

Furthermore, Alexander reasons about the use of his language in comparison to the use of the English language. "This language, like English, can be a medium for prose, or a medium for poetry. The difference between prose and poetry is not that different languages are used, but that the same language is used, differently. In an ordinary English sentence, each word has one meaning, and the sentence too, has one simple meaning. In a poem, the meaning is far more dense. Each word carries several meanings; and the sentence as a whole carries an enormous density of interlocking meanings, which together illuminate the whole." (Alexander, 1977 [12], p. xli).

Buschmann, Henney, and Schmidt (2007 [92]) formulate a hypothesis in their work that a pattern language is built up from over several stages. Firstly, pattern stories describe specific examples of the application of patterns in combination. Secondly, the experiences from the stories are abstracted into pattern sequences. Thirdly, numerous sequences of patterns form a pattern language. These show that the patterns can be combined in a way that helps engineers to solve problems with different solutions.

6.5.2. A Template for Describing a Pattern Language

Note that the difference between a natural language and a pattern language is that a natural language focuses on communication. In contrast, a pattern language focuses on complex engineering activities. Complex engineering problems are often split up into sub-problems, which are addressed separately. Different patterns contain problems and solutions for the different granularity levels of a problem (its sub-problems). Hence, the solution to a design problem often requires the combination of different patterns to be applied in sequence. Moreover, in a pattern language there often exist multiple solutions to a problem, which means multiple pattern sequences. A pattern language contains all these sequences of patterns.

We propose the following template for describing a pattern language:

Patterns (Vocabulary) "The elements of this language are entities called patterns. Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice." (Alexander, 1977 [12], p. x). Moreover, patterns have to be presented in a consistent format "For convenience and clarity " (Alexander, 1977 [12], p. x). The same format also makes them easier to understand and browse.

Patterns have to describe how a solution solves a problem. This solution has to be described in a way that makes it possible to decide if this solution creates an added value (or a benefit) for the user of the pattern. Hence, the engineer can decide if the solution does create the added value he/she is looking for or if the solution should not be implemented to save time and resources. The pattern reveals its meaning (semantic) in the connection between problem and solution.

Note that Alexander does not state in one precise sentence that problem-solution pairs are an element of a pattern language. However, Alexander states (Alexander, 1977 [12], p. x) that a pattern language consists of patterns and in turn that a pattern contains

the essence of a problem-solution pair. Moreover, Alexander states that all patterns of a pattern language should be described using the same format. This format is the so called pattern form. In the following page (Alexander, 1977 [12], p. xi), Alexander states that: "There are two essential purposes behind this format. ... Second, to present the problem and solution of each pattern in such a way that you can judge it for yourself, and modify it, without losing the essence that is central to it." (Alexander, 1977 [12], p. xi). Hence, it is our understanding that patterns in a pattern language should have the same format (the pattern form) and the problem-solution pair is essential to to this form, and the patterns described using the form. We conclude that in turn the problem-solution pair is essential to a pattern form used in a particular pattern language, as well.

Connections between Patterns (Syntax) Each solution includes syntax, a description that shows where the solution fits in a larger, more comprehensive design and which other solutions can refine this design. This relates the solution into a network of other needed solutions. For example, a larger solution might be a house for a place people want to live in. The rooms are part of the house and require ways to get light. One way to get light into a room is an electronic lamp. Another way is a candle. "In short, no pattern is an isolated entity. Each pattern can exist in the world, only to the extent that is supported by other patterns: the larger patterns in which it is embedded, the patterns of the same size that surround it, and the smaller pattern which are embedded in it." Alexander (1977 [12], p. xiii). Additionally, Alexander states "There are two essential purposes behind this format. First, to present each pattern connected to other patterns, so that you grasp the collection of all 253 patterns as a whole, as a language, within which you can create an infinite variety of combinations. ..." (Alexander, 1977 [12], p. xi) Hence, relating patterns is inevitable when forming a pattern language and has also to be reflected in the pattern form.

Pattern Sequences (Grammar) The grammar provides the meaning of sequences of patterns. Meaning with regard to patterns is a solution to a problem, which is derived by applying a sequence of patterns. Note that a pattern language allows that different sequences of patterns exist, which all solve the same problem. This is in line with Buschmann et al. (2007 [92]), who state that numerous pattern sequences form a pattern language. This can be compared to a language in which different sentences can have the same meaning, while being syntactically different. For example, the following sentences are syntactically different but have the same meaning; (1) I have not seen the sun in a long time, and (2) It has been ages since I saw the sun. In short, a grammar explains in which places of what sequences a pattern is useful (Eloranta, Koskinen, Leppänen, and Reijonen, 2014 [126]).

In several books regarding pattern languages all possible sequences of patterns are shown in a diagram, such as Eloranta et al. (2014 [126]), Buschmann, Meunier, Rohnert, Sommerlad, and Stal (1996 [91]), Gamma, Helm, Johnson, and Vlissides (1994 [154]), Schumacher, Fernandez-Buglioni, Hybertson, Buschmann, and Sommerlad (2006 [342]), and Schümmer and Lukosch (2007 [343]). In some cases such as (Eloranta et al., 2014 [126]) the number of possible sequences of patterns lead to a large diagram. To address this problem, the authors show only a partial view of the diagram in the book and reference the entire diagram on a corresponding homepage. Thus, the scalability issue of the diagram size can be solved.

6.5.3. Definitions of a Pattern Language used in Software Engineering

We describe the vocabulary, syntax, and semantics of a pattern language for the related work on patterns for software engineering. We focus in particular on the works of Fowler (1996 [147]), Gamma et al. (1994 [154]), and Schumacher et al. (2006 [342]). We consider the works of Fowler,

because he presented analysis patterns for capturing domain knowledge of enterprise systems. His pattern language for analysis pattern has some impact to be precise 223 citations are listed in the ACM digital library[38]. Fowler's analysis patterns refer to the analysis phase of software engineering and support in particular the structured re-use of elicited domain knowledge. His work has the closest similarity to our work concerning the re-use of elicited domain knowledge using context patterns. To the best of our knowledge no work about patterns for re-using elicited domain knowledge for software engineering with more citations exists. This is why we consider Fowler's work. In addition, we decided to consider patterns for the design phase of software engineering with significant impact. The work on design patterns of Gamma et al. (1994 [154]) has a citation count of 4524 in the ACM digital library[39] and we are not aware of a work regarding patterns in software design with a higher citation count. Similar works, for example, the work of Buschmann et al. (1996 [91]) regarding pattern-oriented software architecture, have a lower citation count. Buschmann et al.'s work has 837 citations[40]. We selected the work of Schumacher et al. (2006 [342]) as a representative pattern-based work concerning a specific knowledge area in software design, in this case security. In the future, we are planning to include further work regarding pattern languages, e.g., the previously mentioned work of Buschmann et al. in an extended comparison of pattern languages using the following structure:

Patterns (Vocabulary) Fowler agrees with Alexander that a pattern language requires to have a common way to describe patterns (Fowler, 1996 [147]; Fowler, 2002 [148]). His common way for describing analysis pattern contains a unique name, structural and graphical description, and a textual description of behavior and relations to other patterns. A very similar understanding of how to describe a pattern can be found in Hafiz et al. (2012 [166]), and Fernandez and Pan (2001 [143]).

Even though Fowler does not follow it strictly in his analysis patterns, he identified a meta structure of design patterns: "It is commonly said that a pattern, however it is written, has four essential parts: a statement of the context where the pattern is useful, the problem that the pattern addresses, the forces that play in forming a solution, and the solution that resolves those forces." (Fowler, 1996 [147], p. 6).

Gamma et al. (1994 [154]) also state that patterns need a consistent format in agreement with Alexander. The format of the author's design patterns is defined by a template, which is structured into different sections. For example, every pattern has among others a section for its name, intent, motivation, applicability, solution, consequences, and known uses.

Schumacher et al. (2006 [342]) use a similar template as Gamma et al. (1994 [154]) for their security design patterns. It is interesting to note that the patterns of the authors have no security specific sections in their template such as security goals, for example, confidentiality. This allows the assumption that the template could also be used in a more general sense for non security related design patterns.

Fowler (1996 [147]) refrains from restricting his *analysis patterns* to a fixed form of a single problem-solution relationship in contrast to design patterns. "A fixed form carries its own disadvantages, however. In this book, for instance, I do not find that a problem-solution pair always makes a good unit for a pattern. Several patterns in this book show how a single problem can be solved in more than one way, depending on various trade-offs.

[38] ACM citation count of Analysis patterns: reusable objects models source: `http://dl.acm.org/citation.cfm?id=265172`

[39] ACM citation count of Design patterns: elements of reusable object-oriented software source: `http://dl.acm.org/citation.cfm?id=186897`

[40] ACM citation count of Pattern-oriented software architecture: a system of patterns source: `http://dl.acm.org/citation.cfm?id=249013`

Although this could always be expressed as separate patterns for each solution, the notion of discussing several solutions together strikes me as no less elegant than pattern practice. Of course, the contents of the pattern forms make a lot of sense - any technical writing usually includes context, problem, forces, and solution. Whether this makes every piece of technical writing a pattern is another matter for discussion." (Fowler, 1996 [147], p. 6-7).

To sum up, Gamma et al. (1994 [154]), and Schumacher et al. (2006 [342]) follow a well defined set of sections in their template that describes each pattern. In contrast, Fowler (Fowler, 1996 [147]; Fowler, 2002 [148]) defines the structure of his analysis patterns more abstract. His structure just requires a name and some form of structural and behavioral description.

However, Fowler states in his meta structure for *design patterns* of other authors (introduced above in the syntax part) that: "This form appears with and without specific headings but underlies many published patterns. It is an important form because it supports the definition of a pattern as 'a solution to a problem in context', a definition that fixes the bounds of the pattern to a single problem-solution pair." (Fowler, 1996 [147], p. 6).

The pattern form of Gamma et al. (1994 [154]) describes the problem in several separate sections. The section *Intent* describes the general design problem. The section *Motivation* states a scenario in which the design problem occurs and the section *Applicability* refers to specific situations in which the design pattern is useful. The solutions are described in several sections, as well. The section *Structure* describes the graphical representation of the pattern. *Participants* illustrates the elements in the graphical representation. *Collaborations* states the elements' collaborations and responsibilities. *Implementations* and *Sample Code* illustrate how to represent the pattern in source code.

Schumacher et al. (2006 [342]) consider the sections *Problem* and *Solution* explicitly in their template. The sections are also paired in the sense that the Solution section follows the Problem sections without any section in between. However, several sections refine the solution such as descriptions of *Structure*, *Dynamics*, and *Implementation*.

Overall, design patterns such as the ones from Gamma et al. (1994 [154]), and Schumacher et al. (2006 [342]) seem to follow the guideline of describing one problem and one solution in a pattern. Nevertheless, Fowler (1996 [147]) still embeds a problem solution relationship in his patterns, but not as strict. In some cases he refers to multiple solutions or problems.

Connections between Patterns (Syntax) Fowler (Fowler, 1996 [147]; Fowler, 2002 [148]) states that the relations between patterns are important. Note that Alexander uses the term connection instead of relation. We argue that connection is a synonym for relationship[41] and use the term relationship for the remainder of this chapter. The reason is that software engineering works such as Fowler (1996 [147]), Fowler (2002 [148]), and Schumacher et al. (2006 [342]) use the term relationship, as well.

Moreover, according to Fowler (Fowler, 1996 [147]; Fowler, 2002 [148]) a pattern language is indeed about the relationship between patterns. Fowler dedicates an entire part of his book (Fowler, 1996 [147]) to *Support Patterns*, which define the relationships between organizational patterns for, for example, *Accounting* to software architecture patterns such as the *Layered Architecture* pattern.

In addition, Hafiz et al. (2012 [166]) agree and elaborate that an enumeration of patterns without defined relations among them is just a pattern catalog. Both, Hafiz et al. and

[41] Connection and Relationship are synonyms according to dictonary.com: `http://dictionary.reference.com/browse/relationship`

Fowler, basically adopt the view of Alexander (1977 [12]) towards connections between patterns being an essential part of a pattern language.

Gamma et al. (1994 [154]) state also that the relationship between patterns is important. They even create a figure to illustrating the relations between their patterns. Moreover, a section in their template defines the relations to other patterns.

Schumacher et al. (2006 [342]) dedicate two sections in their template to documenting the relations between patterns. The *Variants* section contains descriptions of variants and specialisations of a pattern. In addition, the *See Also* section in the template references patterns that solve similar problems and patterns that refine the pattern.

In summary, Fowler (1996 [147]), Gamma et al. (1994 [154]), and Schumacher et al. (2006 [342]) agree that relations between patterns have to be defined. However, each pattern language uses different kinds of relations and different ways to document these relations.

Pattern Sequences (Grammar) Fowler uses two different types of patterns: analysis patterns that refer to a particular business domain, and supporting patterns that describe how to apply the analysis patterns. The pattern sequences in Fowler's work can relate different analysis patterns or analysis patterns and supporting patterns. However, Fowler's books do not contain a diagram that shows all pattern sequences, instead the sequences are written in texts of the individual patterns (Fowler, 1996 [147]; Fowler, 2002 [148]).

Gamma et al. (1994 [154]) show diagrams in their books, which contain all possible sequences of their patterns (Gamma et al., 1994 [154])p. 22 . In contrast, Schumacher et al. (2006 [342]) use a taxonomy for security and map their patterns to the respective parts of the taxonomy.

6.6. A Quick Overview of Context Patterns

In this section we give a quick overview (in alphabetical order) of the context patterns we derived so far and which are the basis for our pattern language on context elicitation. The overview just provides an initial impression of the context patterns. But the overview should be sufficient to understand the following chapter. The context patterns are explained in detail in Chapter 8[42].

Note that the Cloud System Analysis Pattern and the Peer to Peer Pattern are not described in detail in this thesis. These patterns are an important part of the thesis of Kristian Beckers (Beckers, 2014 [37]). We refer the interested reader to Kristian's thesis for detailed information on the Cloud System Analysis Pattern and the Peer To Peer Pattern.

6.6.1. Cloud System Analysis Pattern

Intent This pattern can be used to elicit the technical as well as the organizational context of an application or system which realized using a cloud.

Essence Using clouds has become a common solution for scalability as well as economical reasons. But while clouds are easy to use from a technology point of view on the first sight, they have some major and complex implications on the IT landscape and organizational level. Different new stakeholders enter the scene when migrating to a cloud as well as technological aspects which are hidden, like the used cloud stack or the distribution of data centers. Read our Cloud System Analysis Pattern to become aware of these new stakeholders and aspects. Instantiate the Cloud System Analysis Pattern to collect the new context information, which has to be considered when moving to a cloud, in a structured way.

[42] Page 131

6.6.2. Meta Pattern

Intent A context pattern solves the problem of elicitation and structured description of context information for a particular domain. The Meta Pattern can be used to describe a context pattern. It is of course only necessary to describe a context pattern if no context pattern exists for a particular domain.

Essence When conducting software engineering, it might happen that the software engineers have to deal with and understand a particular domain and the context given by the domain. In many cases, the software engineers are not familiar with this particular domain. And there might be reasons that they also have to document the insights they gained about the domain for downstream activities. But as the domain is unknown, the information which should be collected, and the sufficient degree of abstraction and detail for this information is unknown. And there is also no structured way of eliciting and documenting the known information. Deriving a new context pattern, if none is already available, can help to solve the problems in this situation. Read the Meta Pattern to understand how one should derive a new context pattern. The Meta Pattern contains the information about sources that are useful for deriving a context pattern, elements a context pattern should contain, and a process how to derive and validate a context pattern.

6.6.3. Law Pattern

Intent The worlds of software engineering and laws are two distinct worlds with very different wordings, methods, and culture. Nevertheless, software systems have to be compliant to laws. This pattern describes how to extract and document the essence of laws in a way that they can be used in software engineering in general and in requirements engineering in particular. This pattern is applicable for statute law. Hence, it is not directly applicable for case law.

Essence The wording used in the legal domain and the software engineering domain are very different. So are the methods to analyze a system of interest. Nevertheless, legal experts and software engineers have to cooperate as IT systems have to be compliant to the laws of the jurisdiction they operate in. The first problem to face in such a cooperation is to understand the important parts of the laws under consideration and the wording the laws embody. Without such an understanding, the legal and software engineering world cannot be aligned. Read the structural part of our Law Pattern to understand the essence of laws and the context they form. Afterward, instantiate this pattern for all laws which might be of interest. The solution helps you to elicit all essential information within laws in a structured way and not to forget relevant parts. The Law Pattern and the accompanying method also describe how to document the resulting information. While describing a law you also set up different hierarchies of words which are important within the law. The resulting Law Pattern instances and hierarchies are a brief description of the essence of law, enabling a quick understanding what is of importance within a law. Additionally, the information is documented in a way it is usable in software engineering activities, such as requirements engineering.

6.6.4. Law Identification Pattern

Intent The worlds of software engineering and laws are two distinct worlds with very different wordings, methods, and culture. Nevertheless, software systems have to be compliant to laws. This pattern describes how to structure and document information about a system-to-be in a way that it can be used in for a legal analysis in software engineering in general and in requirements engineering in particular.

Essence The wording used in the legal domain and the software engineering domain are very different. So are the methods to analyze a system of interest. Nevertheless, legal experts and

software engineers have to cooperate as IT systems have to be compliant to the laws of the jurisdiction they operate in. One problem to face in such a cooperation is to understand how the system-to-be should be described so that the information is a sufficient case description and therefore facilitates the legal analysis. One specific problem tackled by the pattern is how to bridge the differences in wording of the legal and the software engineering domains. Without such an understanding, the legal and software engineering world cannot be aligned. Read our structural part of the Law Identification Pattern to understand which information about a system-to-be is important for a legal analysis. Afterward, instantiate this pattern for different parts of the description of the system-to-be which might be of interest. The solution helps you to gather all essential information about a system-to-be in a structured way and not to forget relevant parts. The Law Identification Pattern and the accompanying method also describe how to document the resulting information. While documenting a particular requirement you also relate words of the domain of the system-to-be to words of the legal domain. The resulting Law Identification Pattern instances are a brief description of the essence of a system-to-be, enabling a quick understanding what is of importance for a legal analysis. Additionally, the essence is based on information which is used in software engineering activities, such as requirements engineering.

6.6.5. Peer to Peer Pattern

Intent This pattern can be used to elicit the technical context of an application or system which relies on Peer to Peer (P2P) technologies.

Essence Peer to Peer technologies are used to realize distributed systems which have to be scalable, flexible, and adaptable. P2P technologies have strong benefits regarding some requirements, but also negative implications which restrict the fulfillment of other requirements. But as there are many different P2P technologies for different purposes, and P2P architectures are highly complex, it is hard to foresee which consequences the P2P context has on for the actual system-to-be. Read our P2P Pattern to understand the consequences the P2P context bears. Instantiating our P2P Pattern for context elicitation, and you will be guided through the process of setting the actual P2P context, navigating different possible P2P technologies. The benefits as well as liabilities are collected while describing the context. This information helps one to decide on the sufficiency of a selected P2P context for a system-to-be.

6.6.6. Smart Grid Pattern

Intent This pattern can be used to elicit the context of an application or system which is part of a smart grid. The documentation needs addressed by this pattern might be related to standard or legal compliance, and a preparation of a requirements engineering process.

Essence The smart grid is a complex scenario and it is difficult to identify all relevant domain knowledge about it. The smart grid contains several distributed stakeholders which makes communication and building a common understanding inevitable. Moreover, documenting the domain knowledge of smart grids in a way that is easy to understand and does not lack any details is difficult, as well. Read our graphical Smart Grid Pattern and its templates to understand the essence of the smart grid context. Afterward, instantiate these pattern and templates for your particular scenario variation of the smart grid scenario. The solution helps you to elicit all essential information about the smart grid scenario in a structured way and not to forget relevant parts. In addition, the instantiated pattern can be used as documentation as an initial input for, for example, requirements engineering, security and compliance analysis.

6.6.7. Smart Home Pattern

Intent This pattern can be used to elicit the context of an application or system which is part of smart homes which are part of an electricity smart grid. The documentation needs addressed by this pattern might be related to standard or legal compliance, and preparation of a requirements engineering process.

Essence The smart home is a complex scenario and it is difficult to identify all relevant domain knowledge about it. The smart home contains several distributed stakeholders which makes communication and building a common understanding inevitable. Moreover, documenting the domain knowledge of smart home in a way that is easy to understand and does not lack any details is difficult, as well. Read our graphical Smart Home Pattern and its templates to understand the essence of the smart home context. Afterward, instantiate these pattern and templates for your particular scenario variation of the smart home scenario. The solution helps you to elicit all essential information about the smart home scenario in a structured way and not to forget relevant parts. In addition, the instantiated patterns can be used as documentation as an initial input for, for example, requirements engineering, security and compliance analysis.

6.6.8. SOA Layer Pattern

Intent This pattern can be used to elicit the technological elements and their relations which are part of a system-to-be realized as Service Oriented Architecture.

Essence A service-oriented architecture follows the principles of re-use and encapsulations along business processes. And it heavily relies on services which are selected and used on design-time as well as on run-time. Services can be of different kinds of granularity, ranging from complete business tasks to specific, limited functions, and encapsulate diverse technological elements. As a SOA is not a homogeneous system and all parts of it are distributed, it is challenging to identify all important technological elements and their relation to each other. The SOA Layer Pattern helps to find all relevant parts for a SOA-based system-to-be, and the relations between the technology elements of the SOA. The solution helps you to elicit all essential information about the SOA scenario in a structured way and not to forget relevant parts. In addition, the instantiated pattern can be used as documentation as an initial input for, for example, requirements engineering, security and compliance analysis.

6.6.9. SOA Stakeholder Pattern

Intent This pattern can be used to elicit the context of an application or system which is part of or realized as Service Oriented Architecture.

Essence A service oriented architecture follows the principle of re-use and encapsulations along business process. And it heavily relies on services which are selected and used on run-time. These services are often not under control of the users. As consequence, the environment of the system-to-be contains many, sometimes hidden, stakeholders and complex relations between them. And the system-to-be is distributed and relies on inhomogeneous technology elements. The SOA Stakeholder Pattern helps to find all relevant stakeholders for a SOA-based system-to-be, and the relations between the stakeholders as well as the relations between stakeholders and technology elements of the SOA. The solution helps you to elicit all essential information about the SOA scenario in a structured way and not to forget relevant parts. In addition, the instantiated pattern can be used as documentation as an initial input for, for example, requirements engineering, security and compliance analysis.

6.7. Summary

In this chapter we reviewed and discussed the current state of practice regarding context elicitation in research as well as in the industry. We motivated the importance of context elicitation, collected the problems related to context elicitation and surveyed the existing solutions. Furthermore, we discussed our idea of context patterns and a pattern language for context elicitation, and elaborated how our work fits in the existing body of knowledge about context elicitation.

Our main contributions in this chapter are:

- We have highlighted the importance of context elicitation by surveying and discussing statements given in literature on this topic.
- We have discussed the evidences for existing problems in the field of context elicitation in research as well as in practice.
- We have shown that the collected statements and evidence is mainly based on case studies and empirical work.
- We have performed a survey of the existing solutions, identified gaps, and showed that our works on the topic might close some gaps.
- We have motivated our decision for a pattern-based solution for context elicitation, and have shown that this solution is promising by comparing it to a set of requirements for a context elicitation solution which was derived from our literature review.
- We have discussed the necessary building blocks for a pattern language for context elicitation by reviewing widely acknowledged work on the matter in general and in software engineering specifically.

Chapter 7

A Meta Pattern for Context Patterns

In this chapter we introduce and discuss the wording and process for describing a context pattern. Thus, we are talking in this chapter about the meta layer of context patterns as the wording and process for describing a context pattern are on a more abstract level than the context patterns themselves. The wording and process on the meta layer form a meta pattern which, when instantiated, assists one in deriving and describing new context patterns. The aim of this chapter is to clarify the difference between the levels of abstraction, and to assist the reader in getting a deeper understanding of context patterns, how they are derived, how they are used, and how they are related to each other.

In Section 7.1, we discuss how context patterns are derived and described, and how these activities are related to different layers of abstraction. To clarify the nature of context patterns and the Meta Pattern, as well as their usage and relations, we discuss the different layers of our pattern and relate them to other similar fields of research such as (meta) modeling, or ontology building in Section 7.2. Section 7.1 and Section 7.2 deliver genuine, new content. Then, we present our meta model for context patterns in Section 7.3[1]. This section is based on Beckers, Faßbender, and Heisel (2013 [47])[2]. In Section 7.4[3] a process for describing new context patterns, which makes use of the meta model, is introduced. Section 7.4[4] is based on Beckers, Faßbender, and Heisel (2014 [49])[5].

7.1. Deriving and Abstracting Patterns

When talking about patterns, we are also talking about abstraction (Baumgartner and Kohls, 2013 [33]). The knowledge contained in a specific artifact, such as a piece of code, a business process model, or a context description for a domain, is condensed to its general characteristics which can be also found in other artifacts of the same type. In case we find a group of artifacts of the same type, for example different documents describing the context of the same domain for different applications, for which we can derive and abstract some general common characteristics, such as recurring stakeholders, we can form a pattern describing the knowledge in its generalized form which is contained in all specific artifacts. This way, we can find the different single patterns (words) which form the vocabulary (see Section 6.5.2[6]) for a pattern language.

[1] Page 120
[2] The meta model and the method to derive was initially a contribution the author, and revised in discussions.
[3] Page 126
[4] Page 126
[5] The meta process was initially a contribution by the author, and revised in discussions.
[6] Page 103

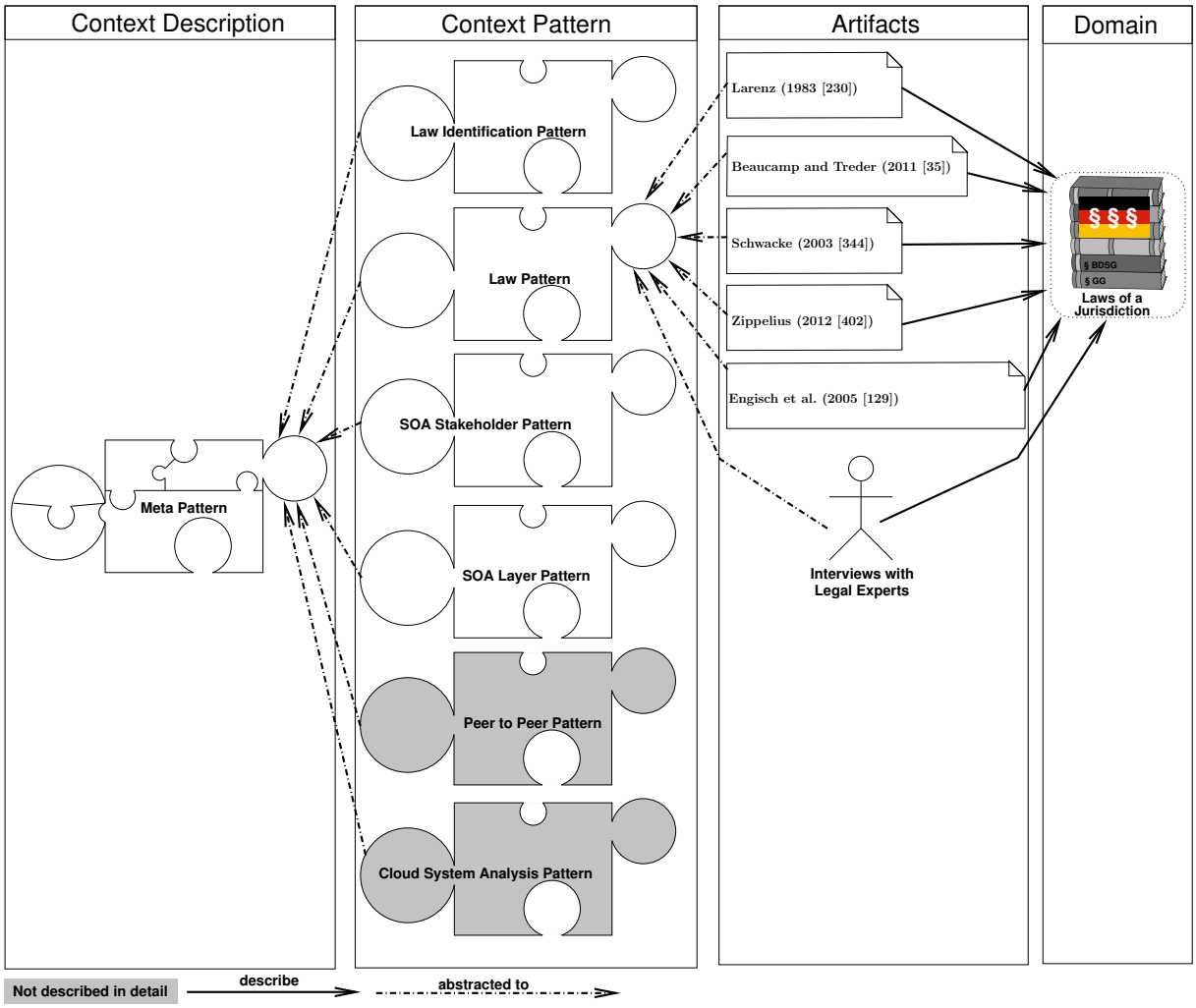


Figure 7.1.: *Patterns and Abstraction*

As already discussed in Chapter 6[7], the patterns of a pattern language should be described in a uniform way to make them easy to understand, comparable and easy to combine. Beside the representation in a unified pattern form, a common vocabulary and representation to describe patterns is commonly acknowledged as helpful (Buschmann et al., 1996 [91]; Buschmann et al., 2007 [92]; Fowler, 1996 [147]; Fowler, 2002 [148]; Gamma et al., 1994 [154]). For example, when talking about design patterns, words with a well established meaning in the software domain, such as class, object, or inheritance, are really helpful to assist the reader in grasping the idea of a pattern at hand. For some domains, such as the software development domain or the business domain, such a wording is already established in different standards and notations. Hence, pattern authors use this wording instead of defining their own. For example, to use the unified modeling language (UML) (UML Revision Task Force, 2012 [374]) or one of its predecessors is very common for patterns in the software engineering domain (Buschmann et al., 1996 [91]; Buschmann et al., 2007 [92]; Fowler, 1996 [147]; Fowler, 2002 [148]; Gamma et al., 1994 [154]). If such a vocabulary does not exist, one should establish it to help the readers understand and apply the patterns correctly. Again, this vocabulary has to be formed out of the commonalities of different patterns. As a result, the patterns are based on a common

[7]Page 81

wording which only has to be understood once. Such a wording can be text only but also a common graphical representation. Such a wording for describing patterns can be regarded as meta pattern.

The elements and relations between the elements used for deriving patterns and the Meta Pattern for our context description case are shown in Figure 7.1. It only shows the legal domain case in detail. For the *legal domain* (column *domain*) it was the aim to understand how legal experts analyze laws and their applicability to a specific case at hand. For this endeavor we analyzed different documents (column *artifacts*) and discussed with several legal experts, who confirmed and broadend our understanding. From these insights we *abstracted* a general method and elements important for describing laws. As a result, we obtained the *Law Pattern* (column *context pattern*). Note that patterns given in gray are not described in detail in this thesis. The other context patterns are described in Chapter 8[8]. A detailed description of the Law Pattern can be found in Chapter 14[9]. In the same manner we derived several other patterns such as the *SOA Layer Pattern*. Those context patterns were distilled in isolation without a common base and wording. Hence, it was not easy to relate and use them together and readers were confused of the different notions used. We aimed at deriving a meta pattern and *abstracted* the context patterns to the Meta Pattern which included the *generalization* of specific elements of the domain specific context pattern to more general context elements valid for all context patterns. Note that the context pattern column does not include the smart grid and smart home pattern. The reason is that these patterns were not used for deriving the Meta Pattern. They were used to validate the Meta Pattern. As a result, we obtained a meta pattern for context patterns.

7.2. Different Pattern Layers

In the previous section we elaborated that dealing with context patterns means using artifacts which differ in their level of abstraction. To clarify the characteristics of these artifacts, which kind information they contain and express, and how they can be used, we align these levels of abstraction to the widely known field of (meta) modeling and ontology building [10].

In the modeling domain it is common to describe the different levels of abstraction with layers (Henderson-Sellers, 2012 [179]). Such a layer hierarchy helps to understand the differences between certain artifacts and helps in the transition between the layers. The layers themselves form a hierarchy from the most abstract to the most specific layer. One of the most prominent layer hierarchies in modeling is the *Object Management Group (OMG) four layer hierarchy* (OMG, 2013 [296]) (see Figure 7.2. The OMG four layer hierarchy is shown in the middle column). The most prominent modeling notation following this layer hierarchy is the UML. The most abstract layer in the four layer hierarchy is *M3*, the *meta meta model layer*. In this layer we find notations which are used to describe modeling notations. The OMG proposes the Meta Object Facility (MOF) (OMG, 2013 [296]) for defining modeling notations. These modeling notations are regarded as *meta models* and contained in the *M2* layer. The UML, or the Business Process Modeling Notation (BPMN) (Object Management Group, 2011 [291]), just to name the most prominent modeling notations, are members of the M2 layer. Such a meta model is then used to produce *user models* which are members of the *M1* layer. The user models are used to describe some data, for example code. Such data forms the *M0* layer. Note that in the beginning the OMG proposed "strict meta modeling" (Atkinson, 1999 [21]), which

[8] Page 131

[9] Page 239

[10] Note that there are other concepts for ordering abstraction levels as discussed by Baumgartner and Kohls (2013 [33]), but as Baumgartner and Kohls (2013 [33]) state they are only applicable in certain cases which do not fit our purpose

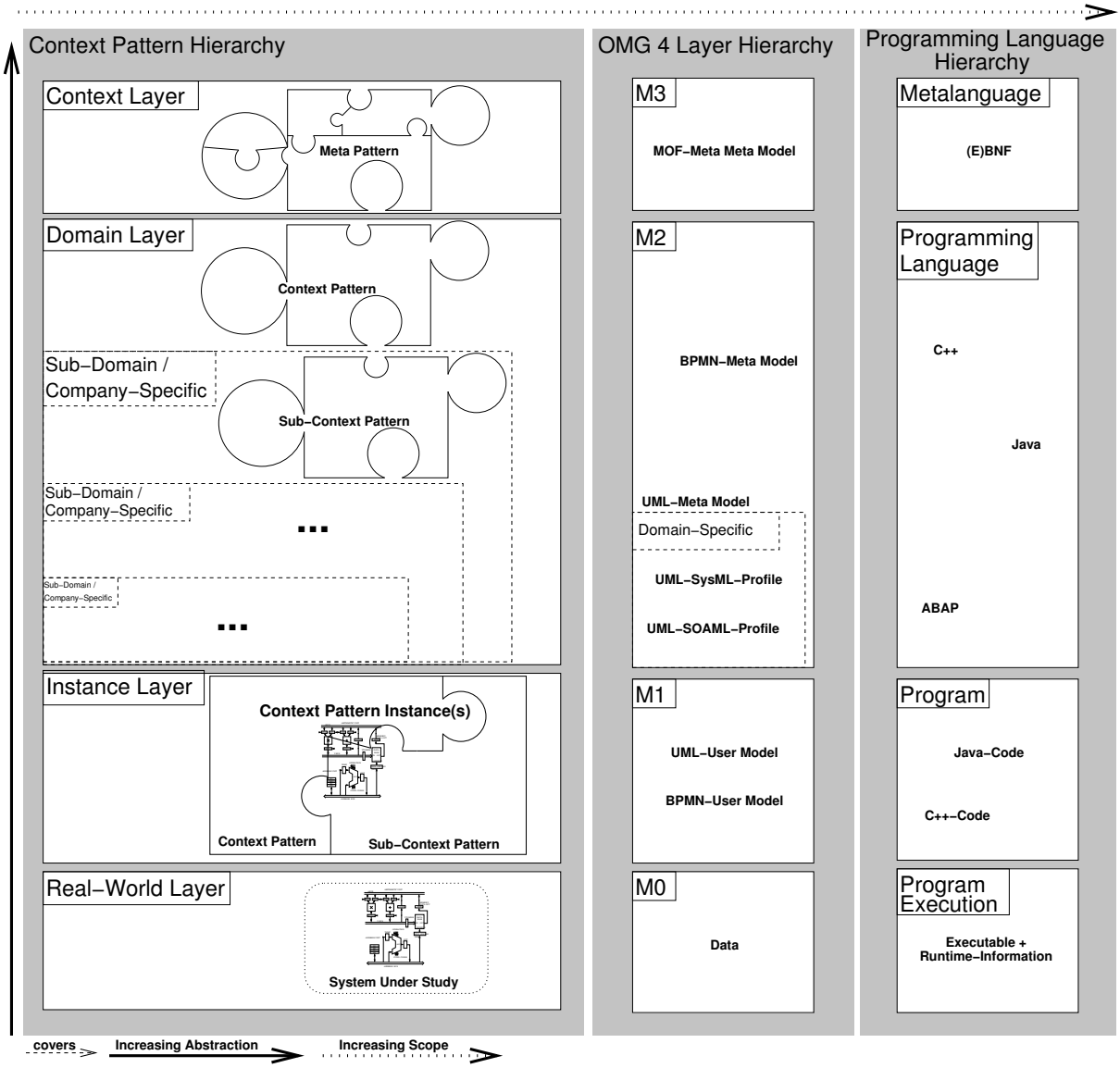


Figure 7.2.: *The Different (Meta)-Layers of Context Pattern (based on Henderson-Sellers (2012 [179]), Bézivin (2004 [55]), and Bézivin (2005 [56]))*

means only those four layers are allowed, and especially forming a (meta) meta model from a (meta) meta model within the same layer is permitted. To some extent these constraints are weakened or ignored in the actual modeling practice. For example, the UML introduced the profile mechanism which allows one to give additional semantic to a UML element. This enables one to build domain specific languages on top of the UML, such as the SysML (OMG, 2012 [295]) for embedded systems, or the SOAML (OMG, 2012 [294]) for service oriented architectures. A more detailed discussion of (meta) modeling and the OMG four layer hierarchy can be found in Henderson-Sellers (2012 [179]).

Henderson-Sellers (2012 [179]), Bézivin (2004 [55]), and Bézivin (2005 [56]) state that also other domains, such as the programming domain, follow the layer principle. Hence, it is a more universal concept not bound to modeling. For example, a *programming language hierarchy* can be mapped to such a layer model (see Figure 7.2. The programming language hierarchy is shown on the right hand side). For example, the meta languages, such as the (Extended) Backus-Nauer

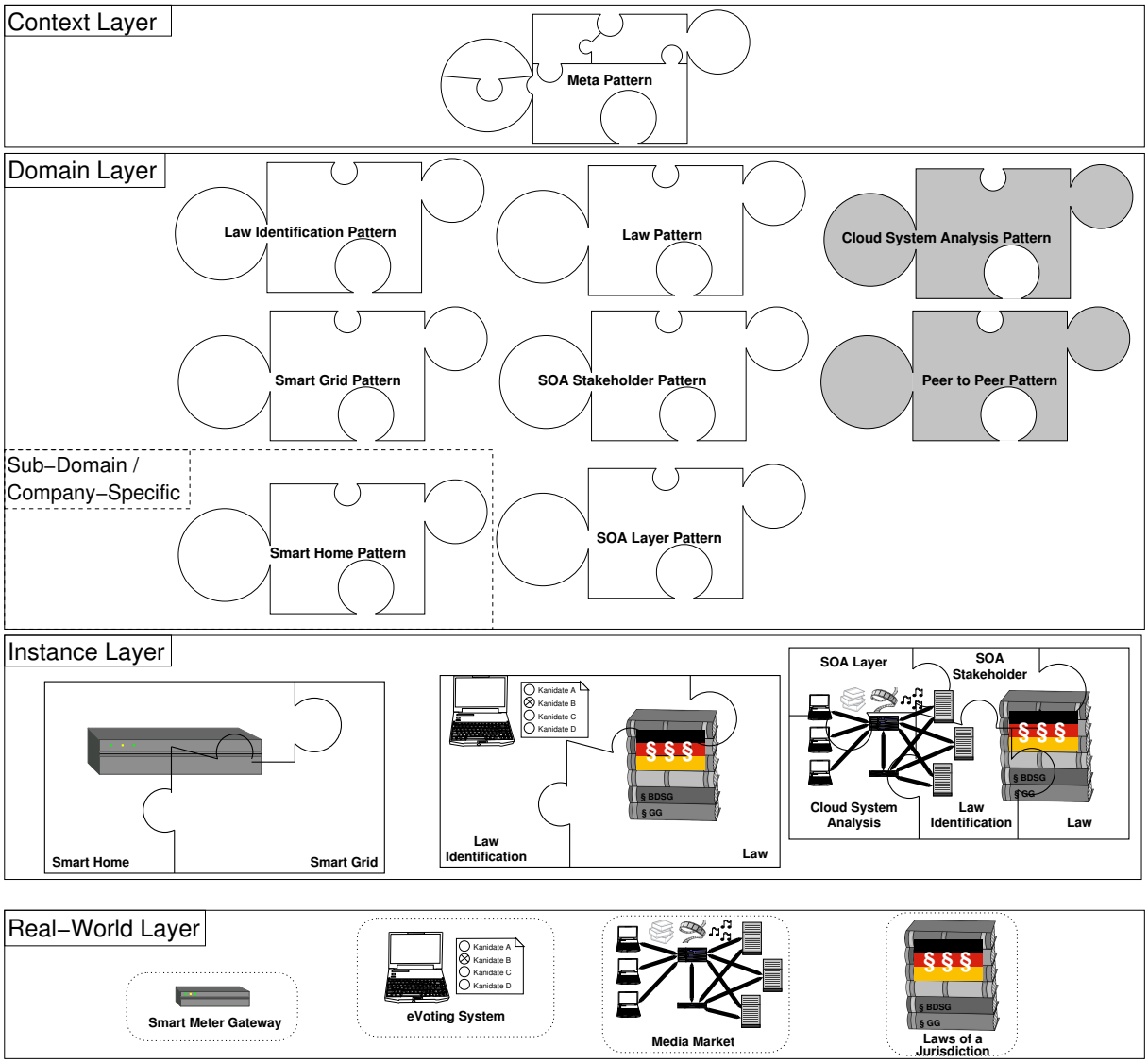


Figure 7.3.: *Mapping of Context Pattern and Their Application to the Layers*

Form ((E)BNF) (McCracken and Reilly, 2003 [263]), are related to a meta meta model in M3. The actual programming languages, such as C++, are members of M2. A program written in a programming language is regarded as user model in M1. At runtime, such a program together with its runtime-information is part of M0.

Indeed, the OMG four layer hierarchy is also applicable to our context description case (see Figure 7.2. The *context pattern hierarchy* is shown on the left hand side). The analogy to modeling and programming might help the reader to grasp the idea and use of the elements on the different levels of abstraction. In the context description case, the most abstract layer is the *context layer*. The context layer contains the *Meta Pattern*, which can be used to describe domain specific *context patterns*. Hence, it is a meta meta model to express meta models in terms of the OMG four layer model. The context patterns are these meta models in our case. They are part of the *domain layer*. And like the UML we are not following the "strict meta modeling" principle and allow to refine a context pattern further to a sub-domain or just exchange the wording to a company specific one. If one or more context patterns are used to describe the context of a specific case, the result are one or more *context pattern instances*. They are members

of the *instance layer* which corresponds to the M1 layer of the OMG four layer hierarchy. And like a user model, a context pattern instance is used to describe something in the real world. Hence, the *described and studied system* is part of the most concrete layer which is the *real world layer*. The context layer, the domain layer, the instance layer, and the real world layer form the context pattern hierarchy which is quite similar to the OMG four layer hierarchy.

Figure 7.3 shows that all patterns and case studies described in this thesis can be mapped without a problem to the context pattern hierarchy. The *Meta Pattern* is the only pattern in the context layer. When instantiated for a specific domain, one gets a domain specific context pattern, such as the *SOA Layer Pattern* or the *Law Pattern*. All those patterns are described in Chapter 8[11]. Exceptions are the Cloud System Analysis and the Peer to Peer Pattern (depicted in gray). One or more context patterns are instantiated to describe a specific system-to-be. For instance, our running example, the *media market*, is described using several Law Identification Pattern instances, Law Pattern instances, a SOA Layer (Stakeholder) Pattern instance, and a cloud system analysis pattern instance. The relations which enable a combination of different context patterns are described in Chapter 9[12]. On the real world layer we find our systems under study, such as the *smart meter gateway*, or the *eVoting system*. Hence, the concept of layer hierarchies is applicable for the context description case, and the analogies to the modeling and programming domain, which might be more familiar to the reader than the pattern domain, might help to better understand the different pattern artifacts.

Note that there are some more relations and differences between the different hierarchies shown in Figure 7.2 which might be confusing. Elements which are part of one hierarchy can be part of another hierarchy. For example, the elements of the program and program execution layer of the programming language hierarchy can be part of the data in the M0 layer of the OMG four layer hierarchy. Hence, we get different descriptions of the same element. A program is described using a programming language, but also parts of it can be described using class or sequence diagrams. Here, we see that those hierarchies cannot be seen as single truth. They are always bound to their focus and purpose. For example, programming languages have the broad scope of describing software and its algorithms. They are applicable to all domains, programming styles, and development processes. The OMG four layer hierarchy and the UML in particular focus on model-driven and object-oriented software development. Hence, it is much narrower and more focused than the programming language hierarchy. But it also gives more guidance how its artifacts should be used. The same happens for the context pattern hierarchy. The context pattern hierarchy is focused on domain-specific context descriptions while the UML is universal to all domains and is able to express more than the context of a software. We use the UML to describe parts of the Meta Pattern and the context patterns. In turn, our pattern models are user models. But still, they are also on different levels of abstraction. The reader should keep in mind that layer hierarchies can be orthogonal to each other even though they have different meta layers. The reader should not be confused when UML diagrams are used for a context pattern as well as the Meta Pattern, which are on different levels of abstraction regarding the context pattern hierarchy, but on the same regarding the UML.

Another domain which is related to the context patterns and the according hierarchy is the domain of ontologies and ontology building. An ontology defines terms and their semantics and relates those terms. In its strictest form an ontology is a tree with "kind of" relations between a parent term and its children. But there are also ontologies which form a graph with various types of relations (see Henderson-Sellers (2012 [179]) for more information on ontologies). An ontology is described using an ontology description language. Hence, every ontology uses a meta language which is similar to a meta model. Indeed, an important part of our context patterns are the terms used. From one layer to the next layer these terms are refined using "kind of" relations.

[11] Page 131
[12] Page 151

For example, the Meta Pattern contains the most general terms, which are then used to define terms of each context pattern. When instantiating a context pattern, we relate these terms to the scenario specific terms. Hence, we also get some kind of ontology when using our context patterns. But our context patterns also contain more information than a pure ontology, such as methods to use them or templates to describe important elements. For a full discussion on the relations between ontologies, (meta) modeling and modeling languages, we refer the interested reader to Henderson-Sellers (2012 [179]).

The discussion on ontologies reveals one important fact about (meta) modeling. A (meta) model only describes structures, but equally important is the use of a (meta) model (Henderson-Sellers, 2012 [179]). This is often overlooked, but when looking at several examples it becomes clear that in practice every (meta) model is accompanied with one or more methods which facilitate the use of this (meta) model. For example, the UML is the central notation used in the rational unified process (RUP) (Kruchten, 2003 [226]). RUP is an instance of the unified process (UP) which is a meta process for creating software development processes using the UML (Jacobson, Booch, and Rumbaugh, 1999 [202]). The UP was created by the same people who formed the UML. The existence of UP stresses the fact that the UML was never meant to be used without a well defined process. Other modeling standards, such as the integrated definition methods (IDEF) (National Institute of Standards and Technology, 1993 [278]) series, directly combine the definition of the notation and the definition of the method to use the notation. Henderson-Sellers and Gonzalez-Perez (2010 [180]) summarize this observation by stating that when people are talking about modeling they are talking about three domains. The first one is the endeavor they want to reach by modeling, the second one is the (meta) model they use for modeling and the third one is the method they apply to use the (meta) model.

Comparing this statement about (meta) modeling to the patterns domain, we see that it is also applicable. We will show in the following that most patterns also cover the domains of endeavor, (meta) model, and method as defined by Henderson-Sellers and Gonzalez-Perez (2010 [180]). A cornerstone of most patterns is the problem-solution pair described in a pattern (Buschmann et al., 2007 [92]). Some examples underpin the importance of the problem-solution relation[13].

Fowler (1996 [147]) refrains from restricting his analysis pattern to a fixed form of a single problem-solution relationship in contrast to design patterns.

> "A fixed form carries its own disadvantages, however. In this book, for instance, I do not find that a problem-solution pair always makes a good unit for a pattern. Several patterns in this book show how a single problem can be solved in more than one way, depending on various trade-offs. Although this could always be expressed as separate patterns for each solution, the notion of discussing several solutions together strikes me as no less elegant than pattern practice. Of course, the contents of the pattern forms make a lot of sense - any technical writing usually includes context, problem, forces, and solution. Whether this makes every piece of technical writing a pattern is another matter for discussion." (Fowler, 1996 [147], p. 6-7).

However, Fowler states in his meta structure for design patterns of other authors (introduced above in the syntax part) that:

> "This form appears with and without specific headings but underlies many published patterns. It is an important form because it supports the definition of a pattern as 'a solution to a problem in context', a definition that fixes the bounds of the pattern to a single problem-solution pair." (Fowler, 1996 [147], p. 6).

[13]Note that the following paragraphs were also part of an earlier version of Beckers, Faßbender, and Heisel (2014 [50]) which was discussed at the EuroPLoP 2014. These paragraphs also appear in Beckers (2014 [37])

The pattern template of Gamma et al. (1994 [154]) describes the problem in several separate sections. The section *Intent* describes the general design problem. The section *Motivation* states a scenario in which the design problem occurs and the section *Applicability* refers to specific situations in which the design pattern is useful. The solutions are described in several sections, as well. The section *Structure* describes the graphical representation of the pattern. *Participants* illustrates the elements in the graphical representation. *Collaborations* states the elements' collaborations and responsibilities. *Implementations* and *Sample Code* illustrate how to represent the pattern in source code.

Schumacher et al. (2006 [342]) consider the sections *Problem* and *Solution* explicitly in their template. The sections are also paired in the sense that the Solution section follows the Problem section without any section in between. However, several sections refine the solution such as descriptions of *Structure*, *Dynamics*, and *Implementation*.

Overall, design patterns such as the ones from Gamma et al. (1994 [154]) and Schumacher et al. (2006 [342]) seem to follow the guideline of describing one problem and one solution in a pattern. Nevertheless, Fowler (1996 [147]) still embeds a problem solution relationship in his patterns, but not as strict. In some cases he refers to multiple solutions or problems.

In general, we can conclude that a pattern describes the problem one can solve by applying the pattern. In terms of Henderson-Sellers and Gonzalez-Perez (2010 [180]) solving the problem is the endeavor. And a pattern also describes how to solve the problem by describing the solution. As we have already discussed in Section 7.1[14], most pattern authors use some notation to describe the structural part of the solution. This structural part could also be called the (meta) model in terms of Henderson-Sellers and Gonzalez-Perez (2010 [180]). And the patterns also describe how to apply the pattern by means of examples or a description of how to implement the pattern. Hence, the (meta) model, the method and the endeavor domain are also part of many patterns.

In consequence, our patterns also cover these three domains. They contain the meta model and the method explicitly as separate parts. Figure 7.4 shows the transition from the context layer to the instance layer of our context pattern hierarchy (Figure 7.2 and Figure 7.3) and the relation to the three domains of (meta) modeling as defined by Henderson-Sellers and Gonzalez-Perez (2010 [180]). The Meta Pattern contains the *meta model* and the *method* to describe a context pattern. In this case, the description of a context pattern is the *endeavor*. As a result, the context pattern (In Figure 7.4 we have chosen the *Smart Grid Pattern* as an example.) is an *instance of* the meta pattern. To describe a sub-domain using an existing context pattern is the second endeavor shown in 7.4. The Smart Home Pattern is described using the Smart Grid Pattern, which provides the meta model and the method. The final endeavor is to describe the context of a smart meter gateway. For this purpose, the Smart Home Pattern is used, which contains the necessary meta model and method. As the Smart Home Pattern is closely related to the Smart Grid Pattern, the context pattern instance for the smart meter gateway is also an instance of the Smart Grid Pattern. This example clearly shows that one should be aware of the layer, which the endeavor is part of, to select the correct patterns to reach the endeavor, and that the domains of endeavor, (meta) model, and method as defined by Henderson-Sellers and Gonzalez-Perez (2010 [180]) are key factors to understand and use a context pattern.

7.3. A Meta Model for Context Pattern

As we have seen in Section 7.1[15] and 7.2, a meta model is one vital part of a context pattern. And the context layer containing the Meta Pattern is the most abstract layer to start with when describing context patterns. In the following, we show in detail how we derived the meta

[14] Page 113
[15] Page 113

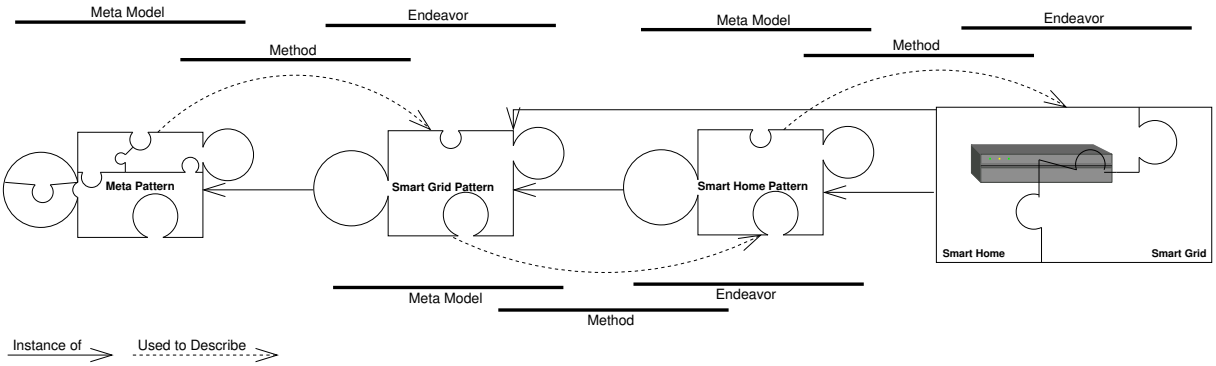


Figure 7.4.: *Using Patterns and the Transition between Layers*

model for our Meta Pattern (Section 7.3.1) and how we validated the resulting meta model (Section 7.3.2).

7.3.1. Deriving a Meta Model for Pattern-Based Context Elicitation

As already discussed in Section 7.1[16], the Meta Pattern, and therefore also the meta model, was derived in a bottom-up way from the different patterns we described independently for different domains. For the process of deriving the general elements, which then form the meta model, we started to analyze each context elicitation pattern in isolation.

Example:
Table 7.1 shows the result of this phase for the SOA patterns. In this case, we analyzed two patterns in conjunction, because the SOA Stakeholder Pattern reuses many elements of the SOA Layer Pattern.

For each element in a context elicitation pattern we discussed what the general concept behind this element is or whether it is a general concept in itself. Several strategies of comparing and abstracting elements can be helpful at this point. Interested readers are referred to Baumgartner and Kohls (2013 [33]) for a discussion of strategies. We used the aggregation strategy which means removing the variations between two elements and checking whether a common, meaningful structure remains or not. Therefore, we set up a table containing the elements of the current pattern to be analyzed as rows and the conceptual elements as columns.

Example:
Table 7.1 is divided in several blocks. The first block contains the general concepts as columns, for example, the concept *layer*. The second and third block contain the elements of the SOA Layer Pattern and the SOA Stakeholder Pattern respectively as rows, for example the element *business organization*.

For each concept found we checked if this concept was already covered in the table or not. In the case it was already covered, we only added a cross to the table. In the case the concept was not covered by a conceptual element, we added a new column and put a cross in it. After iterating over the elements of the pattern, we did a second step by adding the found conceptual elements

[16]Page 113

as rows and analyzing for each of them if they could be further generalized in a reasonable way or not. This way we found also new conceptual elements. Hence, we had to repeat the second step several times.

Example:
The fourth block in Table 7.1 contains the conceptual elements as rows, for example, the *Layer* concept. Conceptual elements added in this step as columns are highlighted by adding a small gap to the first block.

When no new concept was found, we finished the analysis. This way, we obtained the conceptual elements, which were candidates for the meta model.

In a next phase we harmonized the conceptual elements by comparing the found elements, merging them if needed and setting up their relations. This way we got a coherent set of conceptual elements over all patterns. Table 7.2 shows the resulting set of conceptual elements as rows.

In the last phase we had to choose which conceptual elements should be part of the meta model. Table 7.2 shows the conceptual elements and in which of the patterns a corresponding element exists. Additionally, we selected for each pattern those elements which were not explicitly part of the pattern and checked if the missing element is an implicit part of the pattern.

Example:
The SOA Layer Pattern is mainly concerned with the machine. But the layers which are part of the machine are not explicitly mapped to a machine element. Hence, the machine is not explicitly shown. That is why we added a *o* to the corresponding cell in the table (see Table 7.2) row *Machine* and column *SOA Layer Pattern*

The patterns were also tagged with the information if they provide a technical or organizational view, or a combination of both. This is important to consider, because there might be elements which only occur in one of the views. Those elements might be excluded by just looking at the pure occurrence number, because they can only occur in a subset of the patterns. But those elements might be nevertheless important to capture aspects which are specific for a view.

Example:
Indirect stakeholders are only part of patterns which contain also an organizational view (see Table 7.2 row Indirect Stakeholder).

The general rule to include an element into the meta model, was to add every element with an occurrence greater than three, which means the element occurs in more than half of the patterns. In case of a view-specific element, an occurrence of greater than two was sufficient, because the number of patterns associated with a view was four. Every element with an occurrence of two was subject to be discussed. The occurrence of an element was calculated only considering the explicit occurrence in a pattern.

We had to discuss the conceptual elements requirement and machine. For the machine element it seemed that it is only part of patterns which combine the technical and the organizational view. So, the first reason to include them is that for eliciting the context of a software problem the most usual pattern is one which mixes the technical and organizational view. This reason was supported by the experiences of the authors with using context elicitation patterns. A second

		General Concept												
		Layer	Stakeholder	Process	Active Resource	Relation	Environment	Indirect Environment	Indirect Stakeholder	Direct Environment	Direct Stakeholder	Machine	Constituent	Resource
SOA Layer Pattern Element	Business Organizations	x												
	Organization		x											
	Business Processes	x												
	Process			x										
	Business Services	x												
	Business Service				x									
	Infrastructure Services	x												
	Infrastructure Service				x									
	Component-based Service Realization	x												
	Component				x									
	Operational Systems	x												
	CRM				x									
	ERP				x									
	Database				x									
	Packaged Applications				x									
	Legacy Applications				x									
	Participates In					x								
	Performed By					x								
	Relies On					x								
	Exposes					x								
	Business Relation					x								
SOA Stakeholder Pattern Element	Outer System						x							
	Indirect Environment							x						
	Legislator								x					
	Domain								x					
	Shareholder								x					
	Asset Provider								x					
	Inner System						x							
	Direct Environment									x				
	Process Actor										x			
	Business Service Provider										x			
	Infrastructure Service Provider										x			
	Component Provider										x			
	Operational Systems Provider										x			
	Machine											x		
	Influences					x								
	Part Of					x								
	Provides					x								
Conceptual Element	Layer												x	
	Stakeholder													
	Process													
	Active Resource													x
	Relation													
	Environment												x	
	Indirect Environment						x							
	Indirect Stakeholder		x											
	Direct Environment						x							
	Direct Stakeholder		x											
	Machine												x	
	Constituent													
	Resource													

Table 7.1.: *Analysis of the SOA Layer Pattern and the Stakeholder SOA Pattern Elements*

reason was the fact that the patterns with a more technical view contain the machine implicitly. For example, for the SOA Layer pattern the machine is not an explicit model element, but the

extension to the Stakeholder SOA pattern shows that elements of the SOA Layer Pattern directly relate to the machine. We could not find similar evidence for the requirement element. Moreover, we think that the requirement is part of the phases which follow the context elicitation. For the P2P pattern we only added them for visualization means. Law Identification patterns are used in an iterative way. Thus, they are applied after eliciting the ideal context without legal restriction. Hence, this is a very specific case, which one cannot generalize. As a result, the requirement element is excluded and the machine element added to the context elicitation meta model. Finally, we formed the meta model as depicted in Fig. 7.5 out of the selected conceptual elements. The meta model was modeled using the UML notation.

The root element is the *Pattern* itself. Each pattern consists of at least one *Constituent*. In general, an constituent contains elements of the same kind, view or level. An constituent can contain other areas to split it up and make it more fine-grained. An area can be the *Machine*, or an *Environment*, which in turn contains elements that have some kind of relation to the machine, or a *Layer*, which encapsulates elements of the same hierarchy level.

The environment can be further refined. There are elements which directly interact with the machine, captured in the *Direct Environment*. And there are elements which have an influence on the system via elements of the direct environment, captured by the *Indirect Environment*.

An element, which is part of an *Area*, can be a *Process*, a *Stakeholder*, or a *Resource*. A process describes some kind of workflow or sequence of activities. Therefore, it can contain *Activities*. A stakeholder describes a person, a group of persons, or organizational units, which have some kind of influence on the machine. A stakeholder can be refined to a *Direct Stakeholder*, who interacts directly with the machine, and an *Indirect Stakeholder*, who only interacts with direct stakeholders, but has some interest in or influence on the machine. A *Resource* describes some material or immaterial element, which is needed to run the machine or which is processed by the machine and which is not a stakeholder. A resource can be an *Active Resource* with some behavior or a *Passive Resource* without any behavior.

This meta model has several benefits. Some of them were already discussed in detail Sec-

	Type	Technical		Technical, Organizational		Organizational	
	Pattern	P2P Pattern	SOA Layer Pattern	SOA Stakeholder Pattern	Cloud Pattern	Law Identification Pattern	Law Pattern
Meta-model Element	Pattern	x	x	x	x	x	x
	Constituent	x	x	x	x	x	x
	Machine	o	o	x	x		
	Environment		o	x	x	x	x
	Direct Environment		o	x	x	x	x
	Indirect Environment		o	x	x	x	x
	Layer	x	x	x			
	Process		x	x		x	x
	Activity		o	o		x	x
	Stakeholder		x	x	x	x	x
	Direct Stakeholder		o	x	x	x	x
	Indirect Stakeholder			x	x	x	x
	Resource	x	x	x	x	x	x
	Active Resource	x	x	x	x		
	Passive Resource				x	x	x
	Relation	x	x	x	x	x	x
uncovered Elements	Requirements	x				x	
	Requirements leading to P2P	x					
	Requirements Influenced by P2P	x					

x = contains element o = contains element implicitly

Table 7.2.: *Overview of Elements of the Context Patterns and their relation to the Meta model*

tion 7.1[17], and Section 7.2[18]. First, it forms a uniform basis for our context patterns, making them comparable. If a method already makes use of one of the patterns, it is now easy to generalize the usage to the elements of the meta model. This enables one to replace a given used pattern by another one easily. Second, findings and results for one pattern can be transferred to the other patterns via a generalization to the meta model elements. Third, the meta model contains the important conceptual elements for context elicitation patterns. It is helpful to know these elements and search for them in a specific domain when setting up a new context pattern for a domain. Fourth, it enables to form a pattern language for context elicitation patterns (see Chapter 9[19]). The common meta model eases relating the patterns to each other.

7.3.2. Application and Validation of the Meta Model for Context Pattern

After the definition of the meta model, we instantiated it for each of our context patterns. Thus, we aligned all of the patterns to the same foundation, making them comparable. Additionally, when integrating context elicitation patterns into requirements engineering methods, this can be done in general only referring to the context elicitation meta model.

To check whether the meta model is applicable to any context elicitation pattern, we instantiated the meta model for all of the source context patterns, and additionally a smart grid context pattern. This pattern was not part of the set of patterns used for deriving the meta model. Thus, it did not influence the meta model. But in case the smart grid pattern can be fully mapped to the meta model, we show some evidence that the meta model is reasonable and useful in general.

For a structured elicitation of information about the context of a smart grid software, we derived a context pattern for smart grids. For further information on the Smart Grid Pattern see Section 8.2[20]. The resulting elements and their mapping to the meta model is shown in Fig. 7.6.

The root *Pattern* element is the *Smart Grid Pattern* itself. The Smart Grid Pattern contains a *Direct* and an *Indirect Environment*, which map to the meta model elements with the same name. Further, the Smart Grid Pattern contains three *Areas*, namely the *Grid*, the *Micro Grid* and the *Micro Grid Element*. These areas contain different kinds of *Grid Elements*. A grid element is an *Active Resource*. Grid elements are connected by *Grid Element Relations* which are *Relations*. The direct environment contains different kinds of *Direct Stakeholders*, which are

[17] Page 113
[18] Page 115
[19] Page 151
[20] Page 135

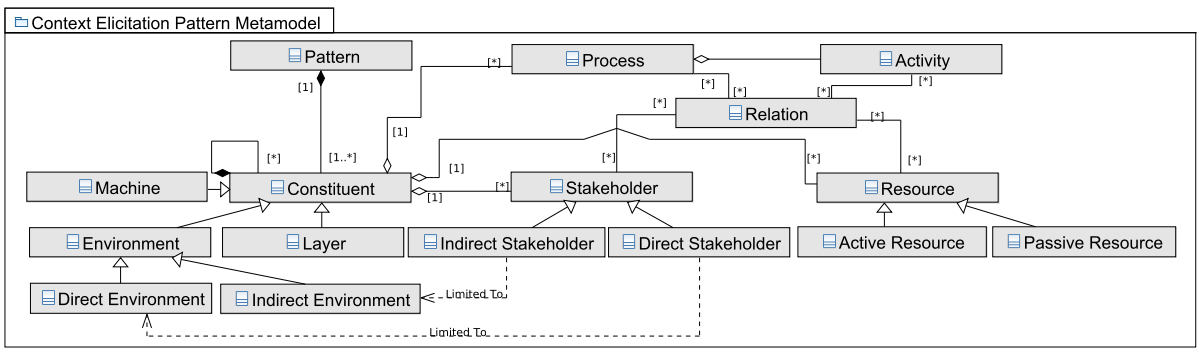


Figure 7.5.: *Meta Model for Context Patterns*

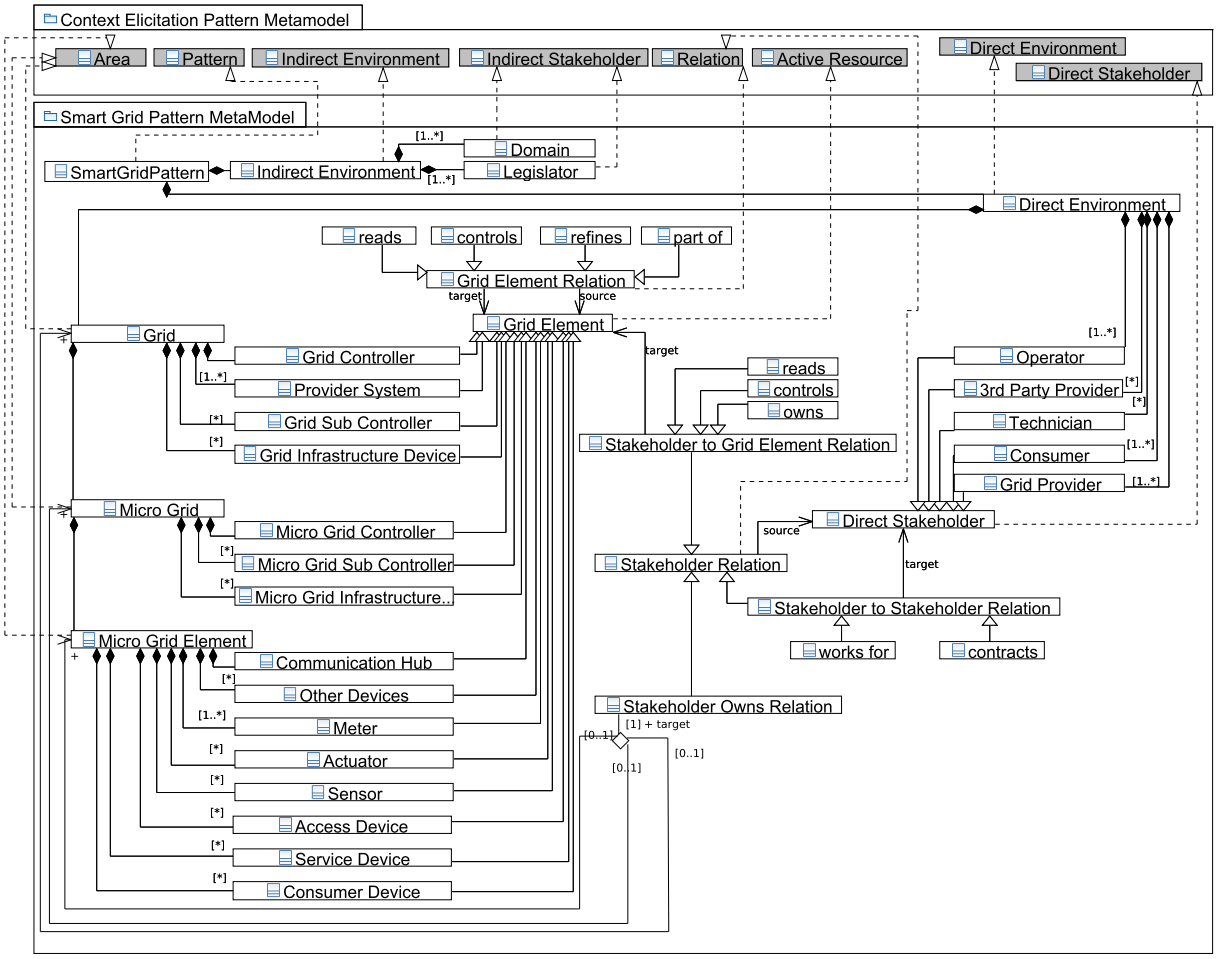


Figure 7.6.: *Smart Grid Pattern Metamodel*

Direct Stakeholders of the meta model. Direct stakeholders are related to each other, to grid elements and to areas by different kinds of *Stakeholder Relations* which are *Relations*, too. The indirect environment contains *Domains* and *Legislators*, which are *Indirect Stakeholders*.

This application of the meta model shows that the meta model is sufficient for describing context patterns. The Smart Grid Pattern could be described using the meta model without any problems. This way we demonstrated that the generalization we did for forming the meta model was reasonable.

7.4. A Meta Process for Deriving Context Pattern

The meta model describes the common entities and relations of context patterns, but it only describes the structural commonalities we identified (Beckers, Faßbender, and Heisel, 2013 [47]) missing the common dynamics, namely the process we used to derive and describe the context pattern. But such a process is also of importance and interest when describing context pattern as discussed in Section 7.2[21]. Hence, we mined protocols, and generated artifacts, as well as intermediate results for commonalities between activities taken. We reflected the insights and compared them to our experience. The result is a meta process (see Fig. 7.7).

The method is conducted by *pattern / IT experts* (short pattern experts) and *domain experts*

[21] Page 115

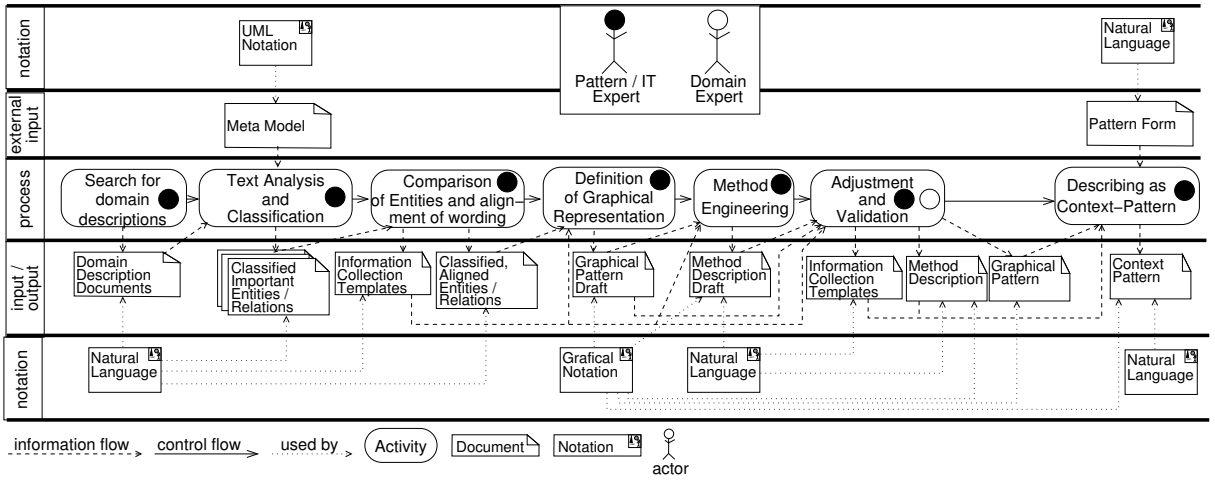


Figure 7.7.: *A process for deriving and describing context pattern*

(see Fig. 7.7). From our experience, it is not efficient to start the collaboration between the pattern and domain experts right from the beginning. The reason is that without any common ground discussions are often fruitless. And the willingness of domain experts to collaborate when one has nothing to provide as a starting point is very small. Additionally, the pattern experts can hardly judge or ask subsequent questions to the answers and results they obtain. As a consequence, the pattern experts should get a basic understanding of the domain and derive preliminary results from freely available documents describing the domain.

Hence, the method starts with *searching for documents* describing the unknown domain. Such documents can be domain standards, technical documents, or scientific or white papers. There should be as many *domain description documents* as possible, which are central for a domain and which need to be of good quality. At the very least, three documents (In the style of the "Three occurrences make a pattern" thumb rule of the pattern community) should be considered in the subsequent steps.

Example:
For the smart grid we used the documents Open Node D1.3 (OPEN node project, 2011 [299]), the protection profile for a smart meter BSI (2011 [76]), and several case studies provided by an industrial partner who was part of the NESSoS project[22].

Once these documents are collected, the text and contained figures have to be analyzed for important entities. At this point, the *meta model* can guide the pattern expert. Reoccurring terms can be collected and mapped to the meta model. This way, a first impression of the basic semantics of a term is captured and entities are formed. Moreover, the meta model gives guidance for which terms one should search.

Example:
Using the meta model, we, for example, classified the entities "other meter managers" OPEN node project (2011 [299]), "supplier of the commodity" BSI (2011 [76]), "producer of the commodity" BSI (2011 [76]), "External entity" BSI (2011 [76]), and "third Parties" (case studies) as important entities.

[22] http://www.nessos-project.eu/

After all entities and relations are collected from the documents in isolation, the *classified important entities / relations* have to be *compared to each other and the wording needs to be aligned.* Entities with the same semantics are grouped and for each group one specific term representing this group is selected. Grouping elements is comparable to the course of abstracting different occurrences to the concept they share. Hence, one can use different abstraction strategies as described by Baumgartner and Kohls (2013 [33]). We used a combination of segregation and aggregation strategy here, which means that we to remove all detailed information which is not relevant for a comparison in a first step. In a second step we removed all attributes which are not shared for the elements at hand and checked whether a common, not trivial structure (such as everything has a name) remains. This way, a coherent set of *classified and aligned entities and relations* is derived from the different documents describing the domain under investigation.

Example:
Comparing the definition and use within the original documents for the classified entities "other meter managers", "supplier of the commodity", "producer of the commodity", "External entity, and "third Parties', we discovered that they have the same semantics. Hence, we choose the term 3rd party provider to represent them in the Pattern.

While classifying the entities, one should also track which information is given about the entities in the source documents. Information which is always given or requested for an entity in all source documents is described using so called *information collection templates.* These templates serve later on, when the context pattern is used for context elicitation, as kind of questionnaire to collect information about an entity. We also tried to discover the commonalities between the different information collection templates. Only for stakeholder entities we were able to find and describe them (see Table 7.3). All other entities are too diverse to describe a shared template beside name and description.

Next, the classified and aligned entities and collaborations between them are *expressed and visualized using a graphical representation.* The graphical representation should only capture the most important information about a domain to keep it understandable. All other information should be stored in the according information collection templates. From our experience, the graphical representation is the central means to support communication, discussion, and information collection. Iterating over the graphics and discussing all entities and collaborations, which are visualized as relations, is really helpful to get a common ground and understanding in the first place. Additionally, the graphics is one means to assure completeness in the sense that no entity is forgotten in the end. In addition, we noticed that discussions are more structured and less abstract when people have something to point at and work with. For the graphical representation, we use the UML-oriented diagrams[23]. The reason is that many people are familiar with this notation. For us, the UML was a reasonable choice, but one may select what ever suits him / her best. For example, when working with the US-military one would probably select an IDEF-like notation (e.g., IDEF1 (National Institute of Standards and Technology, 2014 [279]) or IDEF4 (National Institute of Standards and Technology, 2014 [280])). In the end, one should have a *graphical pattern draft* regardless which notation is used.

The next activity, *method engineering*, is also of great importance as discussed in Section 7.2[24]. In the beginning, we used context patterns without a proper structured method. As a consequence, the process of context elicitation was difficult, as we were sometimes missing information

[23]Note that in some cases we introduced changes to the UML notation in order have an easier to understand graphical representation.
[24]Page 115

Table 7.3.: *Information Collection Template Pattern for Stakeholders*

Name What is the name or identifier of the stakeholder?

Description Which important properties does the stakeholder have? What characterizes the stakeholder? What is its place in the environment?

Motivation Which objectives does the stakeholder follow? Why does the stakeholder influence other entities?

Top Level Goals Which top level goals does the stakeholder have?
☐ **Adaptability** ☐ **Compliance** ☐ **Economy** ☐ **Efficiency** ☐ **Evolvability**
☐ **Learnability** ☐ **Maintainability** ☐ **Modularity** ☐ **Performance**
☐ **Portability** ☐ **Privacy** ☐ **Reliability** ☐ **Resilience** ☐ **Re-Usability**
☐ **Robustness** ☐ **Safety** ☐ **Scalability** ☐ **Security** ☐ **Testability**
☐ **Understandability** ☐ **Usability**

Kind
☐ **Specific** Is the stakeholder a real entity? Is the stakeholder not used to represent a group?
☐ **Representative** Is the stakeholder a real existing entity? Is this stakeholder used as proxy for a group of homogeneous stakeholders?
☐ **Group** Is the stakeholder not a real existing entity? Is this stakeholder used to describe a group of homogeneous stakeholders?
☐ **Role** Can this stakeholder be shared through groups of heterogeneous stakeholders? Are there well-defined rights and permissions for this stakeholder?

for an entity at hand, which we should have collected beforehand. Or information was collected several times, and the coherence of the collected information was hard to maintain. To solve these problems, we started to define methods for collecting the information. We use the agenda principle as described by Heisel (1998 [178]). An agenda describes a sequence of activities with necessary inputs, the desired outputs, actors, the notations used, and validation conditions to check coherence. We accompanied the purely textual representation using agenda diagrams. Figure 7.7 is an instance of such a diagram. One can use what ever he / she wants to describe processes. For us, the agenda principle turned out to be useful. The output of the activity method engineering is a *method description draft*.

All drafts should be *adjusted and validated* in discussions with domain experts. Up to this point, the pattern experts only worked on the basis of documents, which introduced the risks of misunderstanding, incompleteness, and biased results due to the influence of the pattern experts. To discover such issues, it is indispensable to get a review by domain experts. But as the there is already a common ground to discuss and inputs to be reviewed, conducting this activity is not a big issue, nevertheless some iterations might be needed. The final versions of the *information collection templates*, the *method description*, and the *graphical pattern* are the result of this activity.

Last, all final results are compiled to a context pattern description using a *pattern form* (see Section 8.1 for an example). How to actually write a pattern is a topic own its own. We recommend to have a look at Kohls (2011 [223]), Kohls (2012 [224]), and Meszaros and Doble (1997 [267]) Note that the process itself can also be used when someone only wants to understand a specific domain. In this case, he/she might not define a pattern in the end. But from our point of view, the benefits of having a reusable context pattern outweighs the effort

for distilling a pattern.

7.5. Summary

In this chapter, we have introduced the meta model and the meta process which are part of the Meta Pattern for describing context patterns. Beside the representation in a unified pattern form, a common vocabulary and representation to describe patterns is commonly acknowledged as helpful (Buschmann et al., 1996 [91]; Buschmann et al., 2007 [92]; Fowler, 1996 [147]; Fowler, 2002 [148]; Gamma et al., 1994 [154]). Our Meta Pattern assists one when searching for and describing a new context pattern, and it also forms a common base for understanding, relating and combining different context patterns.

But having a Meta Pattern for context patterns also means that we have patterns on different levels of abstraction. To help the reader to understand the different levels of abstraction, the use of the patterns on different levels of abstraction, and the important parts for navigating between different levels of abstraction, we discussed the different levels of abstraction for context patterns specifically and patterns in general under the light of the domains of (meta) modeling and ontologies.

Our main contributions in this chapter are:

- We described how we abstracted from several descriptions of a domain to a context pattern for this domain, and how different context patterns served then as basis for deriving the Meta Pattern.
- We clarified and explained the different layers of abstraction when dealing with context descriptions and context patterns.
- We formed and explained a context pattern hierarchy and related this hierarchy to the domain of (meta) modeling and ontologies.
- We motivated the importance and need of a meta model and method within a context pattern.
- We showed how the layers of interest and the transition between these layers are bound to the endeavor one wants to reach.
- We defined a meta model for context patterns. We explained in detail how we derived this meta model.
- We validated the meta model for context patterns.
- We introduced the meta process (method) for deriving context patterns.

Reading the chapter enables a better understanding of the context pattern contained in the following context pattern catalog.

Chapter 8

A Catalog of Context Patterns

In this chapter, we present a catalog of context patterns. As there is no need for the reader to know each and every pattern in full detail to understand this work, only some context patterns are included in this chapter. The rest of the context patterns can be found on Appendix B.1[1]. The catalog gives an overview of context patterns which are already available and ready to use. Each context pattern is described in detail using the meta model (see Section 7.3[2]) for an aligned wording. The importance of such a wording was already discussed in Section 7.1[3]. We have chosen the Smart Grid Pattern and the SOA Layer Pattern as examples which are included in this chapter, because these two patterns are of particular importance for understanding the following chapters.

The context patterns are presented in a pattern form, which is introduced in Section 8.1. The context pattern form was reviewed and discussed at the EuroPLoP 2014, and published in Beckers, Faßbender, and Heisel (2014 [49])[4]. In Section 8.2, we present our context pattern for smart grids. It was published in Beckers, Faßbender, and Heisel (2014 [49])[5]. The Smart Grid Pattern was refined to a Smart Home Pattern which focuses on end customers' homes. The resulting Smart Home Pattern is part of the Appendix (Appendix B.1.4[6]). In this particular appendix, we will also discuss the differences between the Smart Home Pattern and the Smart Grid Pattern. The Smart Home Pattern is based on Beckers, Faßbender, Heisel, and Suppan (2014 [51])[7]. The second context pattern presented in this chapter is the SOA Layer Pattern (Section 8.3[8]). A second context pattern for the SOA domain is the SOA Stakeholder Pattern. (Appendix B.1.5[9]). These two patterns are used jointly in most cases. The SOA Layer Pattern and the SOA Stakeholder Pattern were published in Beckers, Faßbender, Heisel, and Meis (2012 [41])[10]. Section 8.4[11] concludes this chapter.

Some further context patterns are part of the appendix. In Appendix B.1.1[12], we present our Meta Pattern, which is based on the already introduced meta model (see Section 7.3[13]) and meta process (see Section 7.4). The Meta Pattern was published in Beckers, Faßbender,

[1] Page 396
[2] Page 120
[3] Page 113
[4] The pattern form is a contribution of the author.
[5] The Smart Grid Pattern is a contribution of the author.
[6] Page 422
[7] The Smart Home Pattern was created in several workshops and discussions. The result of this discussions was documented and used to create the Smart Home Pattern by the author.
[8] Page 144
[9] Page 436
[10] The SOA (Layer / Stakeholder) Patterns are a contribution of the author.
[11] Page 150
[12] Page 396
[13] Page 120

and Heisel (2014 [49])[14]. Appendix B.1.2[15] and Appendix B.1.3[16] introduce the Law Pattern and the Law Identification Pattern. These two sections are based on Beckers, Côté, Faßbender, Heisel, and Hofbauer (2013 [45]), Beckers, Faßbender, Küster, and Schmidt (2012 [43]), Beckers, Faßbender, and Schmidt (2012 [44]), Faßbender and Heisel (2013 [133]), and Faßbender and Heisel (2014 [134])[17] We will discuss these patterns and their usage in detail in Chapter 14[18], Chapter 15[19], and Chapter 16[20].

Currently, there exist two more context patterns. The Cloud System Analysis Pattern and the Peer to Peer Pattern. As for these two patterns Kristian Beckers is the main author, we refer the interested reader to his thesis (Beckers, 2014 [37]), or the original publications Beckers, Côté, Faßbender, Heisel, and Hofbauer (2013 [45]), Beckers, Faßbender, and Schmidt (2012 [44]), Beckers, Schmidt, Küster, and Faßbender (2011 [39]), and Moyano, Beckers, and Fernandez-Gago (2014 [274]) for the Cloud System Analysis Pattern, and Beckers and Faßbender (2012 [38]) for the Peer to Peer Pattern.

8.1. A Pattern Form for Context Patterns

Our pattern form is an adaptation of different existing pattern forms
(Fowler, 1996 [147]; Fowler, 2002 [148]; Schumacher et al., 2006 [342]; Gamma et al., 1994 [154]) (see also the discussion in Section 6.5.3[21]). Our pattern form consists of three parts. First the summary, which gives a quick overview of the pattern at hand, second the motivation, which clarifies in detail why and when to use the pattern, and third the solution, which presents the details of the pattern and how to use it.

The summary contains the *pattern name*, which gives a first impression what the pattern is about. The *classification* states if the pattern at hand is on the context layer (see Section 7.2[22]) indicated by the class meta pattern, or if the context pattern at hand is on the domain layer (see Section 7.2[23]) indicated by the classes technical, organizational, and organizational & technical. The latter classes also indicate if the context pattern at hand focuses more on the organizational context of the system-to-be, the technical view or combines both views (see Section 7.3.1[24]). The summary also states the *related patterns*. Note that only those patterns are mentioned explicitly, which can be used to gain further information about the context. For example, for the Smart Grid Pattern the Smart Home Pattern is mentioned, because the Smart Home Pattern contains additional context information whenever the smart grid scenario at hand also includes smart homes. But the Smart Home Pattern does not state the Smart Grid Pattern as related, because the Smart Home Pattern already covers all necessary context from the Smart Grid Pattern. How those relations were established and how to use them is discussed in detail in Chapter 9[25]. The *intent* gives a brief overview when and why to use the context pattern at hand, while the *essence* summarizes the problem tackled by the pattern and how to tackle the problem. The intent and essence enable the reader who is in search for a solution to a problem he / she is

[14]The Meta Pattern is a contribution of the author.
[15]Page 402
[16]Page 413
[17]The initial Law (Identification) Patterns were a contribution of the author, but revised in many discussions. The result of this discussion were fed back by the author to the patterns.
[18]Page 239
[19]Page 267
[20]Page 281
[21]Page 105
[22]Page 115
[23]Page 115
[24]Page 121
[25]Page 151

facing to quickly judge whether it is worthwhile to go into the details of the pattern at hand or not. The last part of the summary enumerates the known uses. The known uses are important as they ground the pattern, which validate the usefulness of the context pattern at hand. To get an overview of the context patterns, it is sufficient to only read the summary of the patterns in the catalog. The summary already transports the basic idea of each context pattern, while the other two parts are of interest when someone is interested in the details of the context pattern at hand.

The motivation starts with an *example*, which shows when and why the pattern was actually used. Such a narrative might help the reader to clarify for him- / herself if her / his problem is similar or not. The example is often regarded as helpful, as readers in some cases have problems to relate their problem with the problem solved by the pattern as the pattern itself is very abstract (Fowler, 1996 [147]; Fowler, 2002 [148]; Gamma et al., 1994 [154]). The *context* abstracts from the example, and describes the general setting in which the context pattern might be used. The *problem* explains in detail which issues might be solved by applying the pattern, and the *forces* explain which influences exist that complicate the problem solving. In the end, the motivation explains in detail when and why to use the context pattern at hand.

The Solution is divided into the structural description of the solution and the method how to use the pattern. They are two fundamental parts of a pattern as already discussed in Section 7.2[26]. The general *structure* is described using a graphical representation, and, whenever the graphical representation does not visualize all important information, complementing textual templates. The *entities* of the graphical representation are explained in detail. In this part, the entities of the context pattern are mapped to the entities of the meta model. This allows the reader to get a quick impression what is described by an entity in general. The *collaboration* part explains the relations between the entities. How to instantiate the pattern is explained in the *method*, which consists of a graphical process description as well as a textual explanation of the method. Both, the structure and the method together form the solution and explain in detail how to apply the context pattern at hand. Our pattern form is shown in the following:

Summary	**Pattern Name**	A unique and descriptive name	**Classification**	Describes the type of context pattern (c.f. Section 7.3.1): **Meta Pattern** (Describes the essence and commonalities of context patterns) **Technical** (Focuses solely on the technical view). **Organizational** (Focuses solely on the organizational view). **Organizational & Technical** (Combines an organizational and a technical view).

[26] Page 115

	Related Patterns	Which patterns are related to the pattern at hand? Patterns given in **bold** are meant to be used jointly with the pattern at hand. Patterns given in *italics* are used as input for the pattern at hand. Pattern given in a normal font can refine elements of the pattern at hand, or elements of the pattern can be refined by the pattern at hand. In most cases both directions are possible at a time. A pattern given underlined is a specialized version of the pattern at hand. (for more information on relations between context patten see Chapter 9)
Summary	**Intent**	Short description when and why to use the pattern.
	Essence	This overview of the pattern presents the problem statements and the essence of the following solution structure in one paragraph. In particular, this part of the solution does not contain any graphical or textual context pattern, but the general way it supports the description of a specific context or scenario. The problem essence is given in normal font,*while the solution essence is given in italics.*
	Known Uses	List of examples where the pattern was applied and from which the pattern was derived.
Motivation	**Example**	A short narrative which exemplifies the context, problem and forces.
	Context	A description of the context where the problem which the pattern solves might occur.
	Problem	A description of the problem.
	Forces	A description of the forces which influence the problem and solution.
Solution	**Structure**	A graphical representation of the pattern in a suitable notation, and complementing templates for capturing information about parts of the graphical representation.
	Entities	Which entities are part of the pattern? A description of those entities.

Solution	Collaborations	What relations between entities exist in the pattern? A description of relations.
	Method	A description how to use the pattern.

8.2. Smart Grid Pattern

We describe in this section the results of our collaboration with domain experts from one of our NESSoS[27] industrial partners in the smart grid domain. We apply our pattern form to describe the Smart Grid Pattern and we also used the Meta Pattern for guidance. The work was done in the context of the EU project NESSoS and is based on real life systems that are discussed in the company. The context pattern was used as a basis of discussion for smart grids and we resolved several misconceptions via discussions based on the context pattern showed in the following.

Summary	**Pattern Name**	Smart Grid Pattern	**Classification**	Organizational & Technical
	Related Patterns	Cloud System Analysis, SOA Stakeholder, SOA Layer, Peer to Peer, Smart Home		
	Intent	This pattern can be used to elicit the context of an application or system which is part of a smart grid. The documentation needs addressed by this pattern might be related to standard or legal compliance, and a preparation of a requirements engineering process.		
	Essence	The smart grid is a complex scenario and it is difficult to identify all relevant domain knowledge about it. The smart grid contains several distributed stakeholders which makes communication and building a common understanding inevitable. Moreover, documenting the domain knowledge of smart grids in a way that is easy to understand and does not lack any details is difficult, as well. *Read our graphical Smart Grid Pattern and its templates to understand the essence of the smart grid context. Afterward, instantiate this pattern for your particular scenario variation of the smart grid scenario. The solution helps you to elicit all essential information about the smart grid scenario in a structured way and not to forget relevant parts. In addition, the instantiated pattern can be used as a documentation, and as an initial input for, for example, requirements engineering, security and compliance analysis.*		
	Known Uses	• CC protection profiles for Smart Meters [76, 77] • The documentation of the OpenNode project [298, 299] • The documentation of the OpenMeter project [297] • The British Smart Grid implementation program [116, 115]		

[27] http://www.nessos-project.eu

Motivation

Example

One application domain of the NESSoS project, which we took part in, is the smart grid. Our task within this project was to deliver solutions to analyze the smart grid regarding security, privacy and compliance, for example with standards, in the early phase of the software development life cycle. The NESSoS industrial partners provided the scenario descriptions. While we are experienced in requirements engineering, security, privacy and compliance in general, we did not have any further knowledge about smart grids. Hence, the results of applying our methods were unsatisfactory in the beginning. The main problem was that we did not speak the language of the industrial partner and we did not understand what we needed to know and how to describe this knowledge. As we also did not know what the important parts of a smart grid are, we could not ask the right questions. The result was a slow, trial and error process annoying both sides. We had to change something to cope with this situation.

Context

Taken from Beckers, Faßbender, Heisel, and Suppan (2014 [51]): "Deriving from the definitions of the European Commission (Commission of the European communities, 2011 [102]), the European Smart Grid Task Force[28], and the Office of Electricity Transmission and Distribution[29], the Smart Grid can be described as a large, flexible, self-monitoring, auto-balancing, and self-regulating electricity infrastructure which uses two-way digital communication to gather and respond on information in an automated manner in order to improve the efficiency, reliability (meaning safety and security), and sustainability of the production and distribution of energy. This new infrastructure will be able to efficiently integrate the behavior and actions of all users connected to it. This means generators, consumers, those that do both, and other third parties that provide services outside of energy generation."

Problem

The general problem of describing the context within a smart grid is centered around different questions:

1. Which elements and information have to be collected to use the context description to judge compliance?
2. Which elements and information have to be collected to use the context description as an input for a basic security and privacy assessment?
3. How to collect knowledge about the important elements of a smart grid in a structured way?
4. How to improve the communication process between the project partners?
5. How to externalize and store the collected information so that it is useful afterward for the different addressees?

Forces

There are several factors which have an influence on the context elicitation, making it complicated to find a solution:

Distributed Infrastructure The smart grid has a distributed nature. As a consequence, the information about the context for a system-to-be might not only reside with the stakeholder of the system-to-be, but also with other players in the smart grid which are not directly concerned with the system-to-be. Hence, cross-organizational knowledge has to be discovered.

[28] `http://ec.europa.eu/energy/gas_electricity/smartgrids/taskforce_en.htm` (last visited on 15-12-2013)
[29] `http://energy.gov/oe/technology-development/smart-grid` (last visited on 15-12-2013)

Motivation **Forces**

Many and diverse standards and regulations The smart grid is a central building block for commodity distribution in the future for many countries. And incidents and issues with the smart grid have a severe impact on societies. Hence, many regulations and standards exist. To collect all these regulations and reflect their impact on the system-to-be is challenging.

Long living systems Smart grids are long living systems whose parts are difficult to evolve. Even though the grid itself is highly flexible regarding the part of taking elements and their interplay, the hardware used is planned to be operating for centuries. Hence, issues in already operating grid elements due to missing or wrong information is a significant threat.

No best practice available The topic smart grid is relatively new. Currently, smart grids are only used on a small scale[30]. Hence, there is no best practice one can follow.

Solution **Structure**

Table 8.3 Information Collection Template for Direct Stakeholders

Name What is the name or identifier of the stakeholder?

Description Which important properties does the stakeholder have? What characterizes the stakeholder? What is its place in the environment?

Motivation Which objectives does the stakeholder follow? Why does the stakeholder influence the organization(s) / provider(s)?

Top Level Goals Which top level goals does the stakeholder have?
☐ **Adaptability** ☐ **Compliance** ☐ **Economy** ☐ **Efficiency**
☐ **Evolvability** ☐ **Learnability** ☐ **Maintainability** ☐ **Modularity**
☐ **Performance** ☐ **Portability** ☐ **Privacy** ☐ **Reliability** ☐ **Resilience**
☐ **Re-Usability** ☐ **Robustness** ☐ **Safety** ☐ **Scalability** ☐ **Security**
☐ **Testability** ☐ **Understandability** ☐ **Usability**

Kind

☐ **Specific** Is the stakeholder a real entity? Is the stakeholder not used to represent a group?
☐ **Representative** Is the stakeholder a real existing entity? Is this stakeholder used as proxy for a group of homogeneous stakeholders?
☐ **Group** Is the stakeholder not a real existing entity? Is this stakeholder used to describe for a group of homogeneous stakeholders?
☐ **Role** Can this stakeholder be shared through groups of heterogeneous stakeholders? Are there well-defined rights and permissions for this stakeholder?

Stakeholder Relations

To	Type	Description
To which target is the stakeholder related?	Which kind of relation?	Detailed Description of the relation.

[30] E.g. http://www.fiercesmartgrid.com/story/honeywell-builds-smart-grid-success-england/2012-01-24

Solution **Structure**

Table 8.4 Information Collection Template for Indirect Stakeholders

Name What is the name or identifier of the stakeholder?

Description Which important properties does the stakeholder have? What characterizes the stakeholder? What is its place in the environment?

Motivation Which objectives does the stakeholder follow? Why does the stakeholder influence the organization(s) / provider(s)?

Kind

☐ **Specific** Is the stakeholder a real entity? Is the stakeholder not used to represent a group?
☐ **Representative** Is the stakeholder a real existing entity? Is this stakeholder used as proxy for a group of homogeneous stakeholders?
☐ **Group** Is the stakeholder not a real existing entity? Is this stakeholder used to describe for a group of homogeneous stakeholders?

Influence

On	Description	Severity
Which direct stakeholder is influenced?	Which kind of influence? What kind of enforcement? What is the base for the influence?	What is the rating for the severity of the influence?

optional entries

Law candidates *(Legislator)* Which laws which might be of relevance for the actual grid (part) to be developed?

Domain-specific regulations *(Domain)* Which domain-specific regulations including, for example, standards and best practice do exist?

Table 8.5 Information Collection Template for Resources

Name What is the name or identifier of the resource?

Description Which important properties does the resource have? What characterizes the resource? What is its place in the environment?

Grid Element Relations

To	Type	Description
Which Grid Element is related?	Which kind of relation?	Detailed Description of the relation.

Solution — Structure

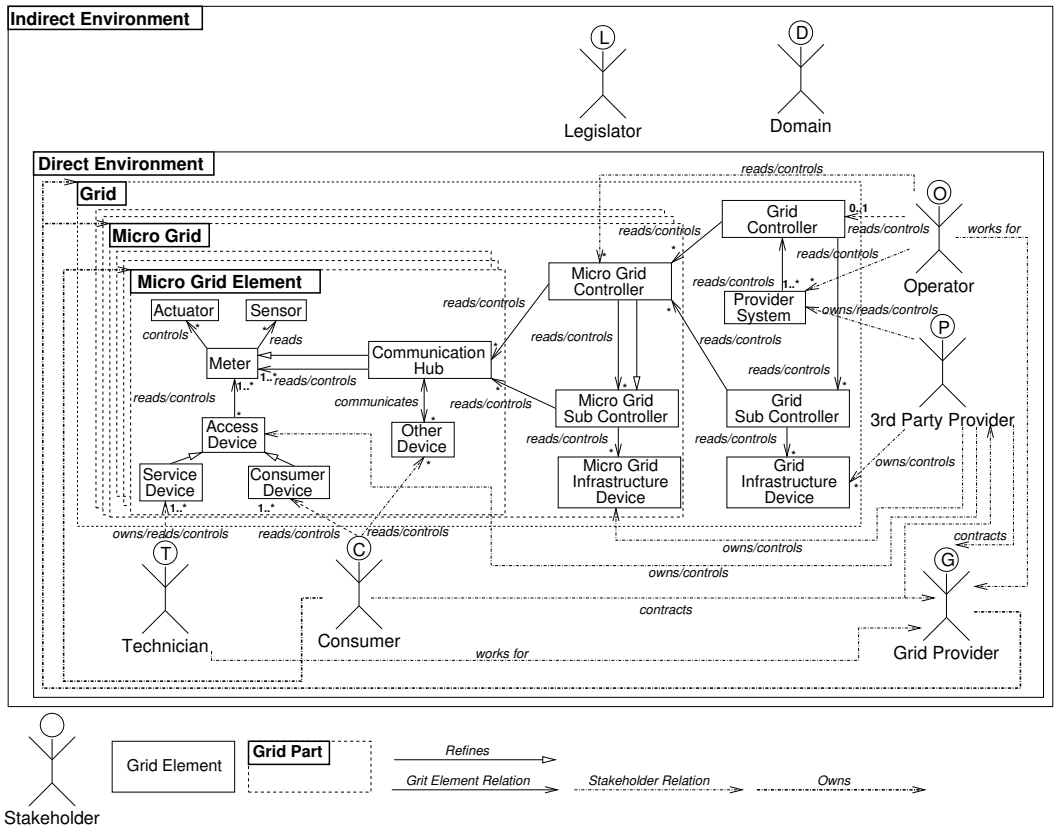


Fig. 8.1: Important entities and relations for the Smart Grid Pattern

Entities — Direct Environment Indirect Environment

Indirect Environment The indirect stakeholders as part of the indirect environment have no influence and, in most cases, also no interest in the machine itself. But they have an influence on the direct stakeholders and therefore they are important for the system-to-be.

Direct Environment The direct environment contains all the direct stakeholders, who have a direct relation to one or more parts of the grid. Hence, they are able to directly influence the grid.

Solution — Entities

Indirect Stakeholder

Legislator The legislator describes the government of a country, for example. A legislator enacts and enforces different regulations which the system-to-be has to be compliant to.

Domain The domain represents the special domains for which the system-to-be is developed. The domain's influence is based on the self regulations of a domain, standards for this domain and so forth. Note that the domain of a smart grid is given by default by choosing the pattern, but there might be more domains of relevance for a problem at hand.

Direct Stakeholder

Operator There are operators for different purposes, e.g., maintenance or billing. Operators work for the the grid provider.

3rd Party Provider Third party providers offer goods or services, which are delivered using the smart grid or are related to the smart grid. The third party providers also have a contractual relation to the grid provider.

Grid Provider The grid provider owns and operates the grid and its major parts. Note that a smart grid can be seen on different scales. Starting from small / local ones to nationwide, to, for example, EU wide. Hence, there can be more than one provider and the characteristics of a provider can also change.

Consumer The consumers have a contractual relation with the grid provider and consume, for example, energy which is provided by the grid or third party providers. Note that this contractual relation can be transitive as a consumer might only have a direct contract with, for example, a third party provider. Nevertheless, there are implicit contractual obligations between consumers and grid providers.

Technician The technicians work for the grid provider installing and maintaining the devices at the consumer side.

Constituent

Grid The ICT-part of the grid is the thing to be built. This does not necessarily mean that all parts of the grid are object of a development project, but at least one will be the machine to be built.

Micro Grid The grid itself is subdivided into micro grids. In simplified terms: A micro grid replicates the grid infrastructure on a smaller scale.

Micro Grid Element The smallest units and part of the micro grid are the *micro grid elements*, for example, households or industrial facilities.

Solution Entities

Active Resource

Grid Controller The central ICT element which coordinates and controls the grid.

Provider System Provider systems are (legacy) systems of a grid or service provider, which have no direct task in controlling the grid, but need access for different purposes, such as billing.

Grid Sub Controller A sub-controller controls one or more micro grids. The sub-controller controls and coordinates the different parts on a more local level. For the overall coordination, the sub controllers are read and controlled by the central grid controller. This controller also allows the direct access of operators or systems of other providers. For smaller grids it is possible that only the grid controller exists and no sub controller are needed for, for example, load balancing.

Grid Infrastructure Device A grid infrastructure device is a device such as power plants or transmission nodes. They are owned and controlled by the grid providers or, in some cases, by third party providers. On the technical side, they are read and controlled by the grid sub controller. Note that a control relation here does not necessarily mean full control, but a (partial) influence on some behavior of the device.

Micro Grid Controller Does the same as the grid controller on local level.

Micro Grid Sub Controller Does the same as the grid sub controller on local level.

Micro Grid Infrastructure Device Is the same as the grid infrastructure device on local level.

Communication Hub The communication hub is itself a smart meter and controls all other meters within a micro grid element. Additionally, it communicates with *other devices* of the element. It is the bridge between a micro grid element and the grid.

Other Device

Meter A smart meter is a means to measure and control the consumption of a certain commodity such as energy.

Actuator An actuator can directly influence its environment. For example, a radiator can heat a room.

Sensor A sensor observes something in the environment such as the temperature or the consumed energy.

Solution

Entities

Active Resource

Access Device An access device is a mean to control and communicate with smart meters.

Service Device An access device which used to maintain smart meters.

Consumer Device An access device which is used by the consumer for retrieving information from the smart meters and controlling the smart meter.

Collaborations

Relation

Stakeholder Relation Relates direct stakeholders and other elements of the smart grid. Three subtypes are possible:

- **Stakeholder to Grid Element** The stakeholder is related to a smart grid element. The relation can be one of the following subtypes:
  - **reads** The stakeholder is able to retrieve information from a smart grid element.
  - **controls** The stakeholder is able to influence the smart grid element.
  - **owns** The stakeholder is the owner of the smart grid element
- **Stakeholder to Stakeholder** The source stakeholder has some relation to the target stakeholder. We distinguish two relations:
  - **works for** The source stakeholder works for the target stakeholder.
  - **contracts** The source stakeholder has a contractual relation to the target stakeholder.
- **owns** The source stakeholder owns all grid elements within the target area which do not have an explicit owner.

Grid Element Relation Relates elements of the grid. Four subtypes are distinguished:

- **reads** The target grid element is read or provides information to the source grid element
- **controls** The target grid element can be influenced to some extent by the source grid element. This includes also transitive control in the sense that the target grid element influences grid elements it controls on command of the source grid element.
- **refines** The target element is refined by the source element. The source element adds new behavior or semantics.
- **part of** The target element is part of the source element.

Solution

Method

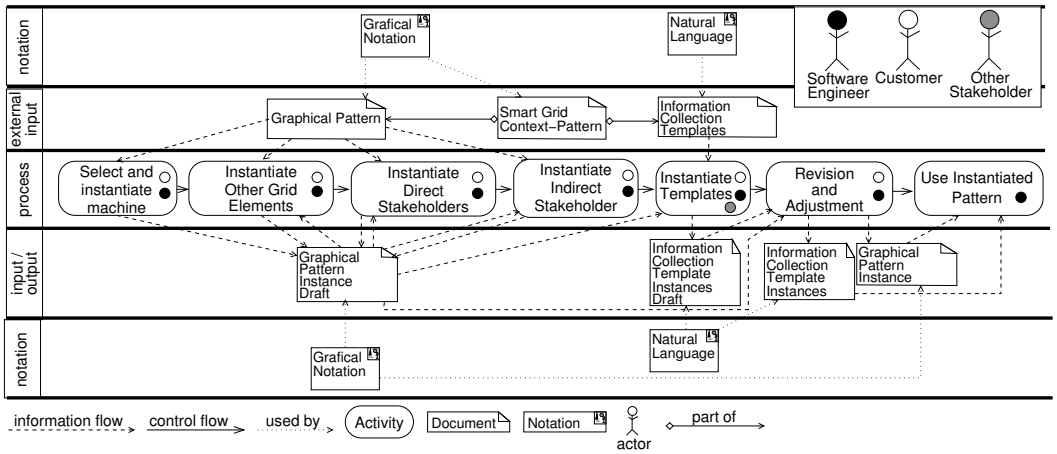


Fig. 8.2: Process Pattern for Using the Smart Grid Context Pattern to elicit the context.

The method starts with the *identification and selection* of the grid element which is the target of the context elicitation. In most cases, this is a single element. In rare cases, it is a set of grid elements or even a complete area. The selected elements have to be instantiated. Next, all *other grid elements* have to be *instantiated*. From our experience, a breadth-first traversal is reasonable. Normally, the elements closely related to the machine are well known and easy to instantiate, while far related elements are not that obvious and often only discovered when the closely related elements are already instantiated. Note that in some cases not all grid elements are of relevance. Hence, they can be left out of the instantiated graphical pattern. When the grid elements are instantiated, one can start to *instantiate the direct stakeholders*. The fact that one already knows the important grid elements eases the instantiation of stakeholders. From our experience, when we start with the direct stakeholders, we often miss one, not having the grid element which is related to this stakeholder in mind. For *instantiating the indirect stakeholders*, one can iterate over the direct stakeholders and reason which domain and legislator are important for this direct stakeholder. When the graphical pattern is completely instantiated, all elements are described by *instantiating the information collection templates*. While for instantiation of the graphical pattern a small number of customer representatives are sufficient, the information collection involves more customer representatives. The reason is that in most cases the detailed information is scattered over different persons. Moreover, detailed information about important grid elements may not even be available within the customers' organization. Hence, one must involve other, external stakeholders. The *final revision and adjustment* should be done with all customer representatives. The result is then *used* by the software engineers for their task. This activity is a complex one, which is a complete process in itself. For example, in Beckers, Faßbender, Heisel, and Suppan (2014 [51]) it is used for a threat analysis.

8.3. SOA Layer Pattern

Pattern Name	SOA Layer Pattern	**Classification**	Technical
Related Patterns	**SOA Stakeholder**, Cloud System Analysis , Peer to Peer		

Summary

Intent

This pattern can be used to elicit the technological elements and their relations which are part of a system-to-be realized as Service Oriented Architecture.

Essence

A service oriented architecture follows the principle of re-use and encapsulations along business processes. And it heavily relies on services which are selected and used at design-time as well as at run-time. Services can be of different kinds of granularity, ranging from complete business tasks to specific, limited functions, and encapsulate diverse technological elements. As a SOA is not a homogeneous system and all parts of it are distributed, it is challenging to identify all important technological elements and their relation to each other. *The SOA Layer Pattern helps to find all relevant parts for a SOA-based system-to-be, and the relations between the technology elements of the SOA. The solution helps you to elicit all essential information about the SOA scenario in a structured way and not to forget relevant parts. In addition, the instantiated patterns can be used as documentation as an initial input for, for example, requirements engineering, security and compliance analysis.*

Known Uses

- IBM reference SOA architecture and method [20, 19]
- SoaML [164]
- Service-Oriented Computing: A Research Roadmap [306, 305]
- A Pattern Language for Process Execution and Integration Design in Service-Oriented Architectures [182]
- A pattern language for service-based platform integration and adaptation [246]
- Patterns: Service-Oriented Architecture and Web Services [127]

Motivation

Example

We had once the situation that a student of ours was also involved in a small size company providing tailored SOA based solutions for customer relationship and enterprise resource management. As it is often the case for start up companies, the development of the product was done in an unstructured, ad hoc manner. And they had chosen SOA for the benefits on the technology level, but were unaware of its implications. They also were missing an overview of stakeholders and their influence on the system. As the customer base was growing and also the usage of the old product, it became obvious to the small company that the established system was not maintainable on the long run as too many requirements were completely overlooked in the beginning. This led to the situation that they planned to replace the old product by a complete rebuild. But they were really in need of a means which helped them to discover all the stakeholders and their relations as well as the technology elements and their relations, and a structured method to integrate the insights in a rigid development method.

Context

For a SOA, the knowledge about the environment and the stakeholders is even more a key to success than for other architectures. Conventional applications are often built for a generic use case, which was obtained by generalizing a set of usage scenarios.

Motivation

Context

To use such an application, the environment has to be adapted to a generic use case. Thus, it often happens that organizations are built around their systems. Changing an organization is costly, and processes that are built to meet IT requirements are often inefficient. In contrast, one major aim of SOA is to enable organizations to build IT systems, which follow their business processes. To reach this aim, it is necessary to be able to adapt a certain single scenario and consider its peculiarities. Hence, the scenario for which a SOA is built has to be described in detail.

Problem

The general problem of describing the context within a SOA is centered around different questions:

1. How to find all technological elements which are either directly or indirectly part of the system-to-be?
2. How do the business processes to be supported look like and how are those processes related to the services to be used?
3. How are the elements of a SOA related?
4. How to collect knowledge about the important elements of a SOA in a structured way?
5. How to improve the communication process between the stakeholders of the different elements to collect the information?
6. How to externalize and store the collected information that it is useful afterward for the different addresses?

Forces

There are several factors which have an influence on the context elicitation for a making it complicated to find a solution:

Rapidly evolving systems As a SOA is process centered it changes according to the processes. And processes tend to change frequently. Hence, it is a problem to be aware of process changes changing the SOA.

Distributed System A SOA has a distributed nature. As a consequence, the information about the context for a system-to-be might not only reside with the stakeholder of the system-to-be, but other players in the SOA which are not directly concerned with the system-to-be. Hence, cross-organizational knowledge has to be discovered.

Dynamic nature of a SOA In a SOA services and the providers of these services can be exchanged easily. But due to these dynamics one must be aware of possible changes and their impact on the system-to-be beforehand and consider them accordingly in the development. Traceability between all abstraction levels is desired for impact analysis.

Only partial control over a SOA Third party services can be used within a SOA but the user gains no control over them. Hence, enforcement of desired properties is more a contractual problem then a development problem. But for certain properties, like privacy, one might not want to hand over the responsibility to someone else. This has to be reflected in the analysis of the system-to-be right from the start.

Solution

Structure

Table 8.7 Information Collection Template for Technological Elements

Name What is the name or identifier of the resource?

Description Which important properties does the resource have? What characterizes the resource? What is its place in the environment?

Technological Element Relations

To	Type	Description
Which technological element is related?	Which kind of relation?	Detailed description of the relation.

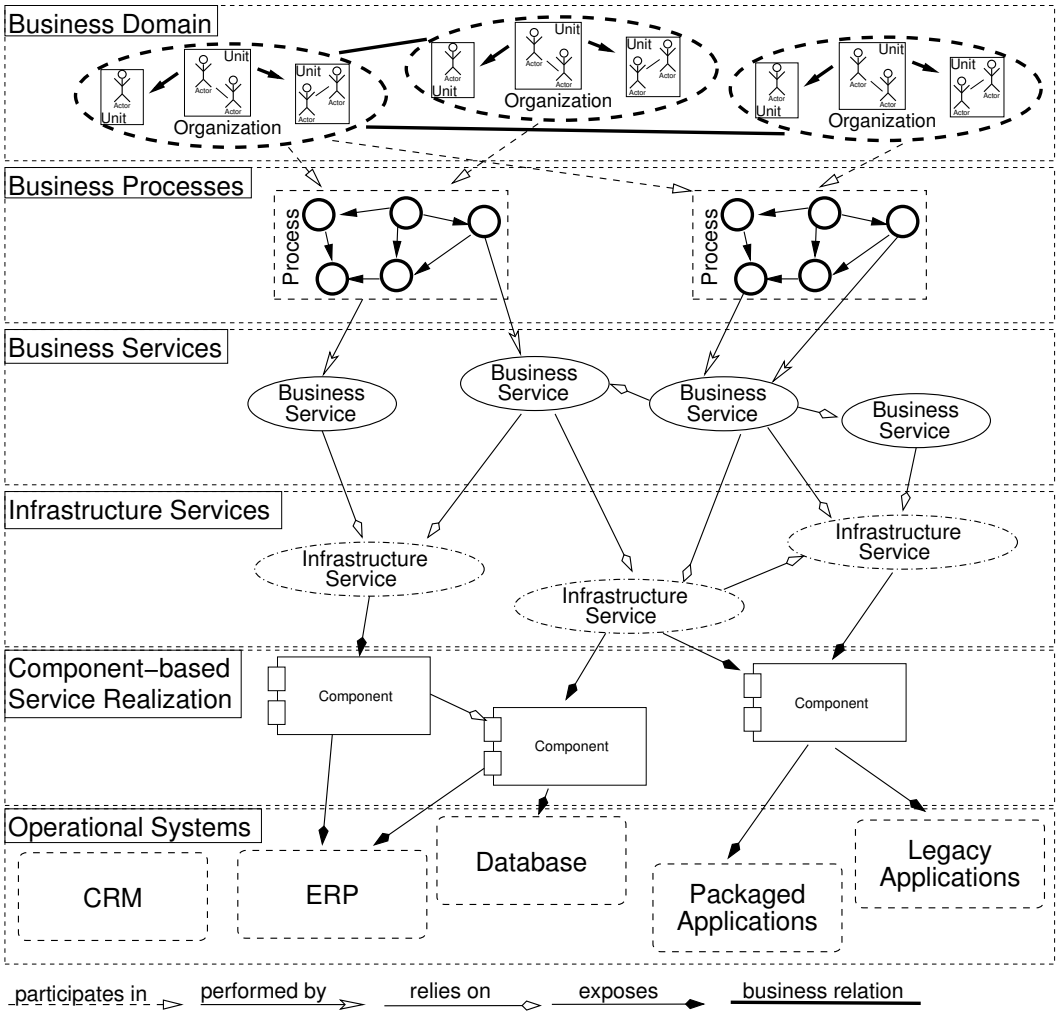
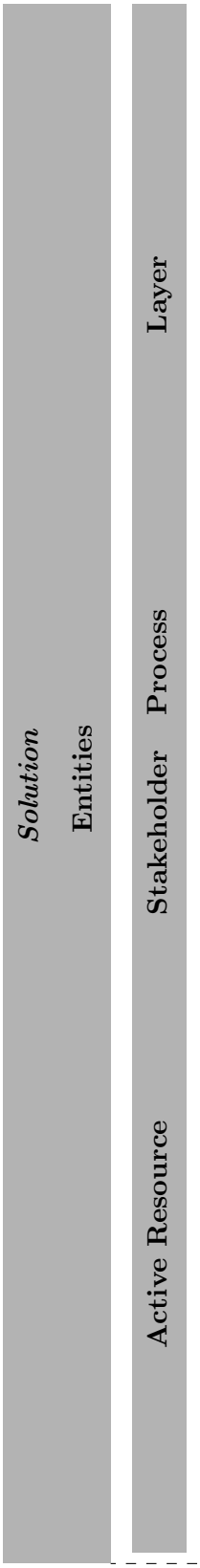


Fig. 8.3: Important technological elements and relations for the SOA Layer Pattern



Business Domain Layer which represents the real world in which the business supported by the SOA takes place.

Business Process Layer which contains the business processes established to run the business.

Business Service Layer which contains the services realizing business functionality.

Infrastructure Service Layer which contains the services realizing infrastructure functionality.

Component-Based Service Realization Layer which contains components used to realize services.

Operational Systems Layer which contains the operational systems needed for running the infrastructure.

Process To run the business, certain processes are executed. Organizations participate in these processes. A process can be executed within one organization or across organizations' borders.

Organizations The business domain consists of Organizations, their structure and actors, and their business relations to each other.

Operational System Operational systems, like databases or legacy systems, are part of the last SOA layer at the bottom of the SOA stack.

Component Encapsulated piece of software which can be integrated in another software to solve a problem.

Infrastructure Service All business services rely upon Infrastructure Services, which form the fourth layer. The infrastructure services offer the technical functions needed for the business services. These technical functions are implemented especially for the SOA or expose interfaces from the operational systems used in an organization.

Business Service Business processes are supported by Business Services, which form the the Business Service layer. A business service encapsulates a business function, which performs a process activity within a business process. Besides atomic business services, there are also composite business services, which rely on other business services. These services are built by composing other business services.

Collaborations

Relation

Business relations A business relation relates two organizations. A business relation is of relevance whenever two organizations collaborate within, for example, processes which are supported by the system-to-be.

Participate in Organizations participate in processes to achieve their business goals or fulfill their contractual obligations.

Performed by A business service encapsulates a business function, which performs a process activity within a business process.

Relies on All business services rely upon infrastructure services to implement their functionality.

Exposes Infrastructure services encapsulate and expose the functionality a component or operational system offers by defining external interface which makes the encapsulated element accessible. In the same way a component can expose an operational system.

Solution

Method

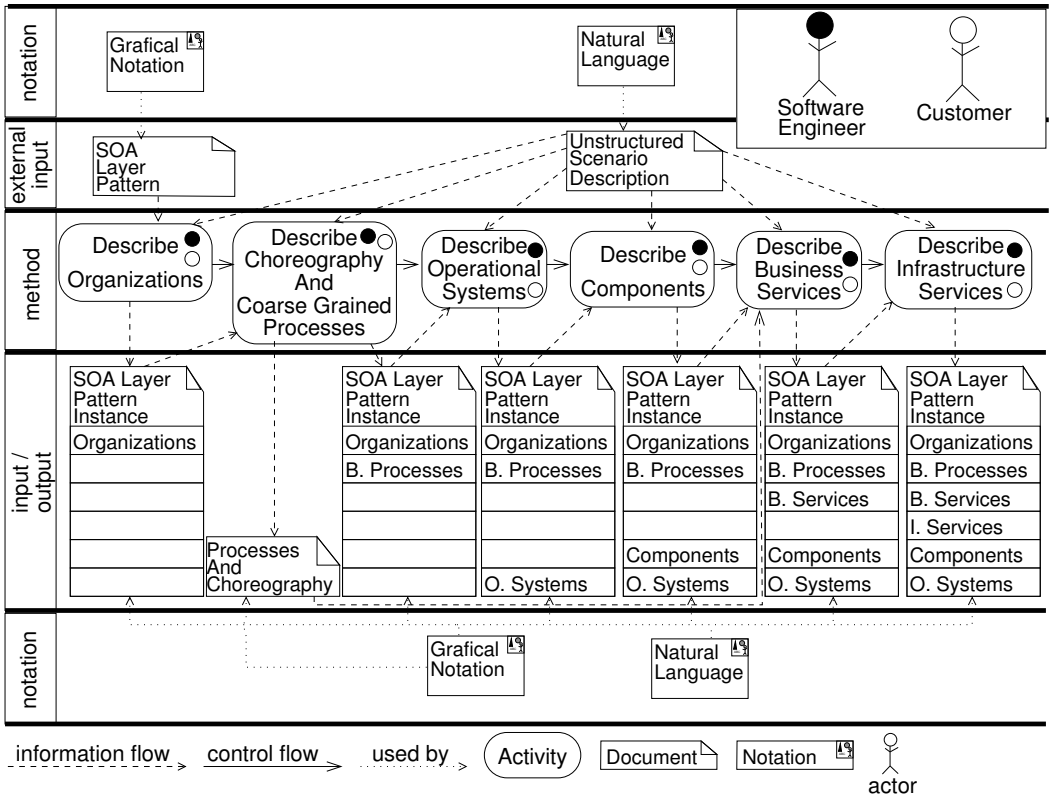


Fig. 8.4: SOA Layer Elicitation

For structuring the information necessary to design a SOA according to the SOA Layer Pattern (see Fig. 8.3), we suggest a meet in the middle procedure. The reason is that the business services and infrastructure services layers are intertwined with the business elements. Therefore, we need both information about the technology-related and the business-related layers. Furthermore, our method provides validity checks for the relations between the different layers.

Solution **Method**

The external input for all steps of this phase (see Fig. 8.4) is the *Unstructured Scenario Description*. The structural part of the, such as the graphical representation and templates, *SOA Layer Pattern* is an additional external input for the first step *Describe Organizations*. It is instantiated layer by layer in separated steps, based on the *Unstructured Scenario Description*.

In the step *Describe Organizations*, we have to collect all relevant organizations. For each such organization we have to collect statements about this organization, describing it further. Next, we have to analyze these statements if they describe business relations between organizations. Last, we have to check for inconsistencies. For example, we have to ensure that no organization is isolated. Being isolated means that there are no business relations to other organizations. In case we find an isolated organization, this organization is either not of relevance for our scenario, or we are missing important information.

In the next step *Describe Choreography And Coarse Grained Processes*, we have to structure the interaction between organizations. Additionally, we structure the available information about internal processes of organizations. The input for this step is the partly instantiated *SOA Layer Pattern*. We start with the organizations and their choreography. The choreography describes the interaction between the organizations. We recommend to document the choreography using an appropriate notation. For example, UML and SoaML collaboration diagrams (Group, 2012 [164]) can be used. UML and SoaML activity diagrams are of use for more detailed interaction and internal process description. The described processes do not have to be complete, but processes and process steps already mentioned or explained in the scenario description should be captured by such diagrams. Next, we have to ensure the coherence of the *SOA Layer Pattern* instance for finishing the step *Describe Choreography And Coarse Grained Processes*. The interactions described by the choreography should reflect the business relations found for the organizations.

For the step *Describe Operational Systems*, we look for statements mentioning IT systems already used within one of the organizations. The operational systems found have to be analyzed for those which are to be replaced, and those which should be wrapped in the new SOA. Only the latter kind of operational system should be added to the *SOA Layer Pattern* instance.

At least one component should be described in the step *Describe Components* for each operational system. Whenever for an operational system there is no statement about a component which should be re-used, the information is missing in the *Unstructured Scenario Description*, or the operational system is unnecessary. Note that components can exist, which are not part of an operational system, but are already mentioned in the scenario description. But to be sure not to miss any relevant operational system, for each component mentioned in the *Unstructured Scenario Description*, we have to check if it exposes such a system.

Up to this point, we have structured the business and the technical parts of the description. Next, we close the gap between those parts. We search for statements, which describe business services for the step *Describe Business Services*. We add those services to the business service layer. For each business service, we have to check if there is a corresponding process step or even complete processes in the processes described in the step *Describe Choreography And Coarse Grained Processes*. If such an activity or process is missing, it should be added. Additionally, we have to check if the business service at hand directly exposes a component, which is already part of the *SOA Layer Pattern* instance. Last we have to check whether the business services at hand is atomic or a composition of other business services. If it is a composition, the business services used to compose it have to be added to the business service layer, too.

Solution Method	The procedure for *Describe Infrastructure Services* is almost the same. One difference is that infrastructure services are mapped to business services instead of activities within the process, and that they can be orphans. It is not necessary that an infrastructure service is used by an already known business service. The reason is that infrastructure services may provide a more general functionality.

8.4. Summary

In this chapter we enumerated different context patterns which form a catalog of context pattern. Such a catalog serves those ones who have some problem when describing the context of a system-to-be, and who are looking for already described solutions in terms of a context pattern. Therefore, each context pattern contains a summary which allows a quick overview of the context pattern at hand. If it might fit the problem one wants to solve, the motivation details the "when" and "why" to use the context pattern. If this meets the needs of the reader he/she can continue with the solution part, which contains all the necessary details for applying the context pattern.

The main contributions of this chapter are:

- A pattern form for context pattern, which allows to get a quick overview of a pattern in the first part of the pattern, while the following parts detail motivation and solution. This assists the reader in getting an overview without unnecessary reading, but also providing the information he/she needs when using a specific pattern.
- A Meta Pattern for deriving and describing a new context pattern in case one does not find a sufficient context pattern in the catalog.
- Domain-specific context establishment based on patterns for the domains
  - Legal
  - Smart Grid
  - Smart Home
  - Service Oriented Architectures
- The context patterns allow a structured elicitation and documentation of relevant stakeholders and technical entities for a system-to-be. Both, the documentation in means of graphical pattern instances and textual template instances as well as the method for collecting the necessary information are explicitly given in each context pattern.
- The context patterns are sufficient for integrating them into software engineering, especially into requirements engineering. This claim will be justified in later chapters (For example, Chapter 12[31], Chapter 16[32], or our work on integrating the Smart Home Pattern into the Microsoft secure development cycle Beckers, Faßbender, Heisel, and Suppan (2014 [51]))
- The patterns are useful beyond software engineering. For example, they can be applied to create the documentation for implementing security standards, e.g., ISO 27001 Beckers, Côté, Faßbender, Heisel, and Hofbauer (2013 [45]), and Beckers (2014 [37]).

[31] Page 213
[32] Page 281

Chapter 9

A Language of Context Patterns

In this chapter, we focus on the relations between context patterns and how to combine context patterns in a meaningful way. These relations form the syntax of a pattern language. The sequences which are used to combine the context patterns form the grammar. For a definition of the terms syntax and grammar in the context of patterns, which differs from the use in other areas such as programming languages, see Section 6.5[1]. We show how one can derive the relations between context patterns using so called pattern relation investigation tables, and which steps one has to take to fill such a table (Section 9.1). This section is based on Beckers, Faßbender, and Heisel (2014 [50])[2]. Additionally, we show how to use a filled pattern relation investigation table to combine context patterns (Section 9.2[3]). This section contains new content. The combination of different context patterns is a common case as the context of most systems-to-be covers different domains. Hence, one needs different patterns to describe the complete context of a system-to-be. Finally, we present the resulting relations between the already existing context patterns and discuss whether we already have reached the state of a full fledged pattern language or not (Section 9.3[4]). This section is based on Beckers, Faßbender, and Heisel (2014 [50])[5].

Section 9.1 is of interest for those who want to provide new context patterns and position them within the existing context patterns. The method described in Section 9.1 helps these *context pattern providers* to find the relations between their new context patterns and the existing context patterns, and how to describe the relations in a way that they are useful afterward for *context pattern consumers*. Section 9.2[6] aims at the ones who want to use and combine different context patterns to a have complete context description for their problem at hand. These *context pattern consumers* get guidance by a method which utilizes the relations and pattern relation investigation tables created by the *context pattern providers*. Hence, the pattern language created and the effort spent once to create it are not an end in themselves, but ease the work of context pattern consumers lowering their effort.

9.1. Deriving a Pattern Language Syntax for Context Patterns

For forming a context pattern language, we investigated the commonalities and differences between the patterns as enumerated in Chapter 8[7]. We also identified how one context pattern can be used in combination with another context pattern. In the following, we will explain the method we used for this purpose. This way, we not only show how we derived our context

[1] Page 102
[2] The pattern relation investigation tables and the according method are a contribution of the author
[3] Page 171
[4] Page 177
[5] The resulting context pattern language and discussion on it is a combined work of Kristian Beckers and the author.
[6] Page 171
[7] Page 131

pattern language to justify it, but also future context pattern providers get some guidance when integrating their context patterns into the existing context pattern language.

9.1.1. Relation Types

Based on our research on the matter of relations between context patterns, we derived relation types which occurred when relating our context patterns. We have identified the following kinds of relations:

specialization of The *specialization of* relation describes that one context pattern was derived from another context pattern. Reasons for having a specialization of a pattern are, for example, a changed wording to reflect the terms used within a company, or that the derived pattern only covers a sub-domain of the original pattern, but contains more details regarding the sub-domain. A *specialization of* relation indicates a close relation between the domains of the two context patterns at hand and the contained elements thereof.

Example:
An example for such a specialization is the Smart Home Pattern which was derived from the Smart Grid Pattern to focus on the details of an end consumer oriented scenario. Details about this example can be found in Section B.1.4.2[8].

The specialization of relation is defined on the *domain layer* (see Section 7.3.1[9] for the layer definition). Figure 9.1 shows the *specialization of* relation from a conceptual view point. When specializing one context pattern to a sub-domain or company specific pattern there are different cases possible what happens to entities of the original context pattern. First, there are entities which are part of both patterns with the same naming (*Entity 2*, *Entity 3* , and *Entity 4* in Figure 9.1).

Example:
The entities *Legislator* and *Domain* are part of the Smart Grid Pattern and the Smart Home Pattern with the same naming.

[8]Page 431
[9]Page 121

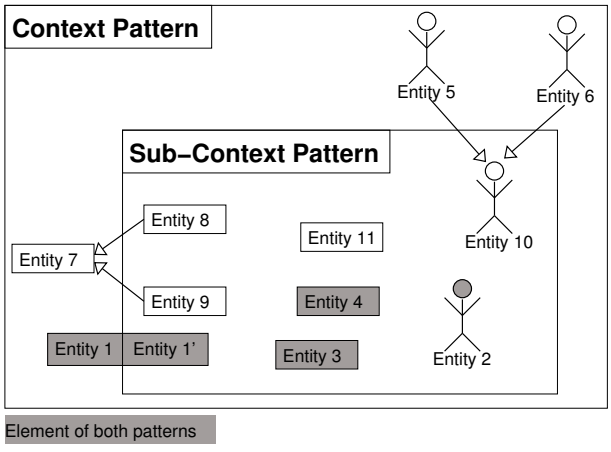


Figure 9.1.: *Specialization of Relation*

Second, there are entities which are semantically the same but named differently (*Entity 1* of the original context pattern named *Entity 1'* in the sub-context pattern in Figure 9.1).

Example:
The *3rd Party Provider* in the Smart Grid Pattern is semantically the same as the *3rd Party Supplier* in the Smart Home Pattern, but a renaming to a company specific wording has taken place.

Third, some entities of the original context pattern are refined to different entities in the sub-context pattern (*Entity 7* of the context pattern is refined to *Entity 8* and *Entity 9* in the sub-context pattern in Figure 9.1).

Example:
The *3rd Party Provider* in the Smart Grid Pattern is not only renamed, but also the entities *3rd Party Service Provider* and *3rd Party Energy Provider* were refined from the *3rd Party Provider*.

Fourth, some entities of the context pattern can be merged as in the view provided by the sub-domain pattern as there is no need to differentiate them (*Entity 5* and *Entity 6* of the context pattern are merged to *Entity 10* in the sub-context pattern in Figure 9.1).

Example:
The entities *Grid* and *Micro Grid* in the Smart Grid Pattern were merged to the entity *Grid* in the Smart Home Pattern as from the perspective of a household it does not matter whether it is part of a grid or micro grid.

Additionally, there might be new entities which are not a reoccurring part of the context as described by the context pattern, but which are an essential part of the sub-context (*Entity 11* in Figure 9.1).

Example:
The *3rd Party Plugin* is a new entity in the Smart Home Pattern as such plugins are specific to smart homes and not occuring in general in smart grid settings.

This relation is:

- *asymmetric* as a context pattern cannot specialize another context pattern which in turn is a *specialization of* the context pattern at hand.
- *irreflexive* as a pattern cannot be a *specialization of* itself.
- *transitive* as all pattern which specialize a pattern at hand also specialize the parent pattern of the pattern at hand.

can refine The *can refine* relation describes that one context pattern refines specific elements of another context pattern. In contrast to the *specialization of* relation the *can refine* relation is defined for the *instance layer*. For example, one context pattern instance contains an element representing services, but this context pattern is not concerned with describing the context of

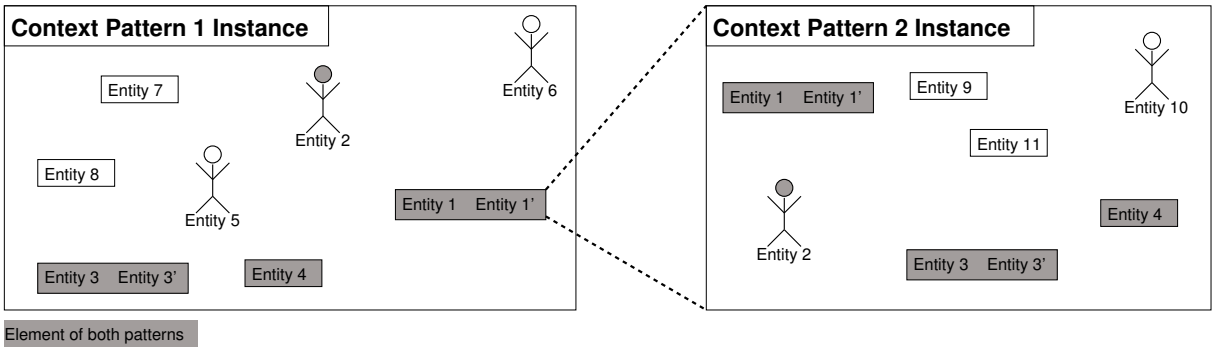


Figure 9.2.: *Can Refine Relation*

these services, while another context pattern specifically describes how the context of these services looks like. Additionally, the semantics or abstraction level of the related elements might differ on the instance level.

Example:
In the following we will assume that we have already instantiated a Smart Home Pattern.

In consequence, it depends on the actual setting whether a refinement is needed as additional context for an element, and whether the actual instance of the element allows the usage of the other pattern. The *can refine* relation is shown in Figure 9.2. It depicts that the entity *Entity 1* which is part of the *Context Pattern 1 Instances* leads to some further context which is described using a *Context Pattern 2 Instance*.

Example:
In one setting the 3rd party plugins of a smart home scenario are instantiated to represent different plugins with varying implementations. In such a case, one might not be interested in the implementation specific context of a plugin. But for our example we assume that we are interested in one specific plugin, which is realized using a SOA. Hence, the SOA Stakeholder Pattern provides some additionally needed context.

Hence, we name the relation *can refine* in order to make it obvious that this relation given between two context patterns may not hold for all possible instantiations of the related context patterns. In case one context pattern is used to refine the context of an entity of another context pattern instance, there are some possible cases how the entities of the already existing context pattern instance are used for instantiating the newly found context pattern. First, some entities might be semantically the same. Hence, their instances are used in both context pattern instances (*Entity 4*, and *Entity 2*).

Example:
The legislators and domains as instantiated for the Smart Home Pattern are the same for the SOA Stakeholder Pattern instance.

Second, some entity instances of the already existing context pattern are also part of the newly found context pattern, but the entity used to represent them is semantically different. Hence,

they need to be mapped from the initial context pattern instance to the newly found context pattern instance. (*Entity 1* and *Entity 3* of the context pattern 1 instance are mapped to *Entity 1'* and *Entity 3'* in the context pattern 2 instance in Figure 9.2).

Example:
The plugin in the Smart Home Pattern is an active resource while it is the machine under consideration in the SOA Stakeholder Pattern. Another example is the Prosumer of the Smart Home Pattern which is mapped to a process actor in the SOA Stakeholder Pattern.

Additionally, both pattern instances can contain entities which are not mapped at all.
This relation is

- *directed* as the fact that a context pattern at hand can refine another context pattern does not imply that a *can refine* relation also exists from the other context pattern to the context pattern at hand. For example, an element which is the machine in one pattern can be refined by another pattern where it is mapped to some resource. But when looking at the other pattern, this resource will never be of central concern. Hence, it cannot be the reason for a refinement of the context. But there are also cases where an entity of a source pattern is mapped to an entity of the target pattern where this entity can be the reason for refinement to the source pattern. Consequently, the relation is not symmetric, nor is it asymmetric.
- *irreflexive* as a pattern cannot refine itself.
- *intransitive* as an entity X which is the reason for a refinement between pattern A and B might not be the one of the entities which might be the reason for a further refinement of B using pattern C. Hence, there is no transitivity.

input The information contained in the instantiation of one pattern can be the input for another pattern. Unlike the *can refine* relation, the *input* relation expresses that the context as described by an existing context pattern instance is extended by the target context pattern not on the basis of a single entity but extends the context in general. Hence, the decision if a target context pattern should be used to extend the context does not rely on the existence of a specific entity instance, but on other circumstances, for example, combinations of entity instances. In consequence, it might happen that several existing entity instances are combined to extend the context using the target context pattern, and that the target context pattern is instantiated several times.(In Figure 9.3 the combination of *Entity 3*, *Entity 2*, and *Entity 4* is used to instantiate *Context Pattern 2 Instance 1*, while the combination *Entity 4*, *Entity 5*, and *Entity 6* is used to instantiate *Context Pattern 2 Instance 2*).

Example:
The decision if one wants to use the Law Identification Pattern does not rely on an specific entity which is part of another context pattern instance, for example a SOA Stakeholder Pattern instance, or not. The decision relies on the big picture given by the whole instance, meaning if there are elements or combinations of elements which indicate the relevance of being compliant in general. As we will discuss in Chapter 16[10], every system has to be compliant, but there might be cases where the developer decides to take the risk of being non compliant based on the information given by, for example, a SOA Stakeholder Pattern instance. When deciding to use the Law Identification Pattern, one does not instantiate it once but several times for different

combinations of elements of the original context pattern instance. For example, we show how the information contained in a Cloud System Analysis Pattern instance can be used as input for the Law Identification Pattern in Beckers, Faßbender, and Schmidt (2012 [44]).

This relation is

- *asymmetric* as the source pattern has to be instantiated before the target pattern. Hence, the target pattern cannot be an input to the source pattern.
- *irreflexive* as a pattern cannot be input to itself.
- *intransitive* as the target pattern might add further entities which are necessary for using the target pattern as an input for further patterns.

used jointly For some of our patterns, we propose to use them jointly. But a *used jointly* relation is just a suggestion, all context patterns can also be also solitarily. A *used jointly* relation is defined on the *domain layer*. It is characterized by the fact that both patterns which are proposed to be used jointly are basically describing the same context but offer a different view on it. The different views are not defined by, for example, a domain versus a sub-domain, but by their focus, for example, a technical view versus an organizational view. Hence, one pattern is not a specialization of the other pattern. Figure 9.4 depicts this relation. *Context Pattern 1* and *Context Pattern 2* describe the same *Context* but focus different entities. Of course, both patterns can also share some elements. Note that the *used jointly* relation always comes along with a *can refine* or *input* relation as these relation kinds explain how two use the two context patterns at hand jointly.

[10]Page 281

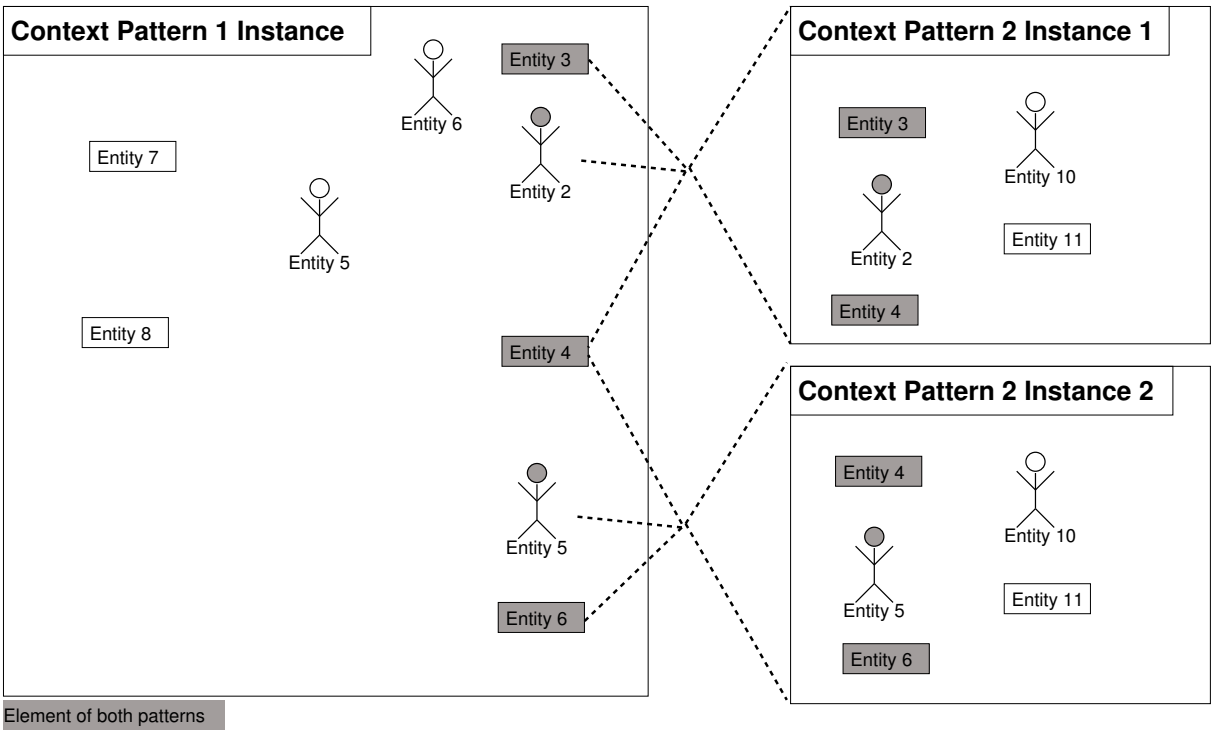


Figure 9.3.: *Input Relation*

Example:
The SOA Layer Pattern, which focuses more on the technical part of a SOA, is supposed to be used together with the SOA Stakeholder Pattern, which reflects more the organizational context. Both describe the SOA context and share some entities, but as they focus on different aspects they also describe entities and relations not part of the other pattern.

This relation is

- *symmetric*.
- *irreflexive* as a pattern cannot be used jointly with itself, which would mean to use the pattern two times for the very same context. It is possible to apply a pattern several times within a broad context which is shared for all instances, but each part of the context described by an instance is slightly different to the other instances. For example, Law Pattern instances might describe the same law, but different sections.
- *transitive*, because patterns which are used jointly describe the same context from different views, and views can be combined freely. Hence, if pattern A can be used jointly with pattern B to combine these two views, and pattern B can be used jointly with pattern C to combine these two views, of course one can use pattern A and pattern C to combine these two views.

Note that we distinguish only these four types of relations. We did not find any additional type while relating our context patterns to each other as described in Section 9.1.4[11]. We explicitly tried to find other relations as described in other pattern languages, for example Gamma et al. (1994 [154]), and Schumacher et al. (2006 [342]).

To make sure that we did not overlook some relation, or that a possible relation just did not show up in our set of context patterns but may occur in general, we assessed our relations under the light of some commonly acknowledged classifications for pattern relations. This enables us to discuss which classes of relations are covered and if there are properties we might have missed for a class we cover. For those classes we currently do not cover, we can check whether this is due to our set of patterns and we missed a potential relation type, or if this type does not apply in general.

Welie and Veer (2003 [386]) and Noble (1998 [286]) have published such classifications. Welie and Veer (2003 [386]) distinguish three fundamental relation types, which can have different variations and flavors:

[11] Page 166

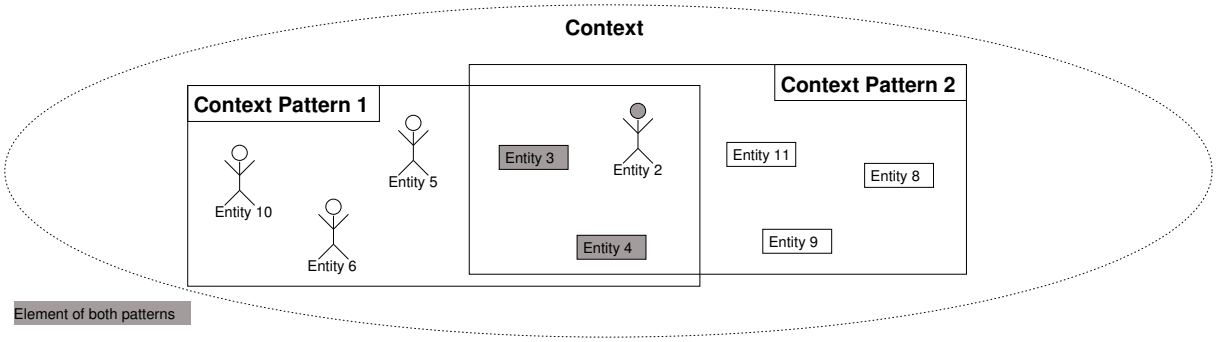


Figure 9.4.: *Used Jointly Relation*

Aggregation One pattern is part of another pattern and completes it.

Specialization One pattern can be derived from (parts) of another pattern and adds more detail or parts of the solution.

Association Patterns with the same context and the same problem to be solved.

According to this enumeration, our relation types for context patterns only cover the specialization relation type. The *specialization of* relation is the direct instance of the specialization relation as defined by Welie and Veer (2003 [386]). The *specialization of* relation could also be seen as aggregation, but in an aggregation all elements of the sub-pattern are also parts of the parent pattern. This is not the case for our *specialization of* relation.

Also the *can refine* and the *input* relation are specializations which are quite special regarding their specialization properties.

The *can refine* relation does not indicate that a pattern is derived from another pattern, but that the pattern which *can refine* the other pattern might add more detail for one or more elements of the original pattern. Hence, exploiting a *can refine* relation always starts at one element of the source pattern, adding more context to this specific element (see Section 9.1.4[12]). Additionally, the refinement relation can be bidirectional, which is quite uncommon regarding Welie and Veer (2003 [386]). This is due to the nature of a context which is always bound to the entity one is exploring. And each context contains entities which add more context if inspected in detail.

The *input* relation also does not indicate that a pattern is derived from another pattern, but an *input* relation adds more detail for a combination of entities of the source pattern. Hence, it does not add additional context to one entity but to a group of entities. For example, the combination of two specific process actors and a specific process described using the SOA Stakeholder pattern might lead to an additional legal context in case they are used to instantiate parts of a Law Identification Pattern.

The association relation can be mapped to some extent to the *used jointly* relation. But the joint use of patterns is not that strict, as the patterns which are supposed to be used jointly share the same context. However, as their view on the context differs, also the problem to be solved differs. For example, the Law Pattern and the Law Identification Pattern are supposed to be used jointly, and both apply for the legal context. But the Law Pattern solves the problem of extracting important elements from a law text, while the Law Identification Pattern is used for structuring requirements in a way that they can be used for a legal assessment. The *used jointly* relation is not an aggregation relation as none of the patterns related are a complete part of the other pattern.

For the aggregation relation we have no counterpart. The reason is that a context pattern for a domain and a specific view on the domain (problem) should be so general that there is no need for further context patterns for this domain and view. Hence, there is no second pattern which could be *completely* part of the first. For example, the *used jointly* relation indicates that two patterns have the same context but their view (problem) differs. As a result, context patterns which can be used jointly only share some elements. Similar reasonings apply for the *can refine* relation and the *input* relation. An exception might be the sub-context patterns, such as the Smart Home Pattern, for which the context and the problem only differ in some details between the domain and the (sub-)context pattern. But as already stated, sub-context patterns are directly derived from context patterns, and therefore the relation is a specialization relation according to Welie and Veer (2003 [386]). Additionally, a sub-context pattern can contain elements the according parent context pattern does not contain, which contradicts the definition of an aggregation relation.

[12] Page 166

Noble (1998 [286]) classified relations between object-oriented design patterns and identified the following main relationships:

Uses A pattern uses another pattern

Refines A more focused pattern refines a general pattern

Conflicts Two patterns address the same problem, but the solutions cannot be applied together at a time.

Our relations for context patterns map to Nobel's design pattern relations as follows. The *used jointly* relation and the *input* relation map to the *uses* relation, but not in the strict sense of Noble (1998 [286]) as one context pattern *only might be* the input for another pattern and two patterns are *only suggested* to be used jointly. In general, all context patterns can be applied without using further context patterns. We have identified two relations that can be mapped to the refines relations of Noble (1998 [286]). The *specialization of* relation is the direct counterpart to the refines relation of Noble. Our *can refine* relation is a focused refines relation as the *can refine* relation only targets some elements of the target pattern instead of the whole pattern. In contrast to Noble's work, we do not have a *conflicts* relation, because each context and view on the context requires a separate context pattern. The reason is that a context pattern for a domain and a specific view on the domain should be that general that there is no need for further context pattern for this domain and view. For instance, a cloud computing scenario requires the Cloud System Analysis Pattern and no other context pattern such as the SOA Layer Pattern can be applied to this particular context.

9.1.2. Tables for Finding Relations between Context Patterns

For investigating the relation between two context patterns we use so-called *pattern relation investigation tables*. We explain the general structure of a pattern relation table in this section. The pattern relation investigation table is of use for *pattern providers* as well as for *pattern consumers*. In Section 9.1.4[13] we explain how a pattern provider can fill and use such a table for two context patterns which shall be related. In Section 9.2[14] we show how a *pattern consumer* can utilize filled pattern relation investigation tables to find the appropriate context patterns for describing his / her problem at hand. The template for a pattern relation investigation table is shown in Figure 9.1.

A pattern relation investigation table compares two given context patterns. In the *first column* of such a table the general element types are stated (like *Constituent* in row 1) as defined by the meta-model discussed in Section 7.3.1[15]. Therefore, a row of a pattern relation investigation table contains the refined elements (like *CP1::Element1* in row 2) which correspond to the given general element type. The columns contain the corresponding elements of the two patterns under investigation. This way, a relation investigation table shows the commonalities and difference of two patterns and allows one to derive the type of relation between two patterns.

Example:
We use the Smart Grid Pattern and the SOA Stakeholder Pattern as an example for relating two context patterns. Table 9.2 illustrates this example.

[13] Page 166
[14] Page 171
[15] Page 121

Row		Only in Context-Pattern One (CP1)	same	Mapping		Only in Context-Pattern Two (CP2)
				Element in CP1	Element in CP2	
1	Constituent	CP1::Element9				CP2::Element10
2	Machine			CP1::Element1	**CP2::Element2**	
3	Environment					
4	Direct Environment					
5	Indirect Environment					
6	Layer					
7	Process					
8	Activity					
9	Relation					
10	Stakeholder			**CP1::Element4**	**CP2::Element3**	
11	Indirect Stakeholder					
12	Direct Stakeholder					
13	Resource			**CP1::Element8**	CP2::Element11	
14					CP2::Element13	
15	Active Resource			**CP1::Element7**	*CP2::Element2*	
16	Passive Resource					

Table 9.1.: *Pattern Relation Investigation Table Template*

Some of the refined elements only occur in one of the patterns at hand. Hence, they are added to the columns *Only context pattern One (CP1)* or *Only context pattern Two (CP2)* (for example, Table 9.1 row 1). These elements are not of relevance when two context patterns are combined.

Example:
For the meta-model type *constituent*, the Smart Grid Pattern and the SOA Stakeholder Pattern are completely disjoint (see Table 9.2 row 1 and columns *Only...*).

Some of the refined elements are semantically the same in both of the patterns. They are added to the column *same*. Later, when one uses the instances of the two different patterns, all refined elements of the *same* column should be synchronized. This means, for example, that adding one instance of such a refined element to one context pattern instance results in adding the instance of the refined element to the other pattern as well.

Example:
The indirect stakeholders domain and legislator are the same for the Smart Grid Pattern and the SOA Stakeholder Pattern. Hence, when adding an instance of a legislator to the Smart Grid Pattern instance the same legislator instance must also be added as a legislator of relevance to the SOA Stakeholder Pattern instance (see Table 9.2 row 13 and column *same*).

Other elements do not have exactly the same semantic in each of the patterns but can be mapped to elements of the other pattern. These mappings are not one to one in all cases. Some elements can be mapped to more than one element of the other pattern (see Table 9.1 row 13 and row 14). In this case, it depends on the actual scenario which element is used for the actual mapping.

Example:
The direct stakeholder 3rd party provider can be mapped to an organization in case the provider does not control a service or an operational system in the scenario under investigation. If the 3rd party provider offers a service or operational system, the provider might be an operational

Row		Only in Smart Grid Pattern (SGP)	same	Mapping		Only in SOA Stakeholder Pattern (SSP)
				Element in SGP	Element in SSP	
1	Constituent	Grid, Micro Grid, Micro Grid Element				Inner System, Outer System
2	Machine			***Grid Controller***	**Machine**	
3				***Provider System***		
4				***Grid Sub Controller***		
5				***Micro Grid Controller***		
6				***Micro Grid Sub Controller***		
7				***Other Device***		
8	Direct Environment	Direct Environment				Direct Environment
9	Indirect Environment	Indirect Environment				Indirect Environment
10	Layer					Business Domain, Business Process, Business Service, Infrastructure Service, Component-Based Service Realization, Operational System
11	Process					Process
12	Relation	Stakeholder Relation (Stakeholder to Grid Element[reads, controls, owns], Stakeholder to Stakeholder[works for, contracts], owns), Grid Element Relation(reads, controls, refines, part of)				Influence, Part Of, Participates In, Provides
13	Indirect Stakeholder		Legislator, Domain	*3rd Party Provider*	Organizations	Shareholder, Asset Provider
14				*Grid Provider*		
15	Direct Stakeholder			Operator	Process Actors	Component Provider
16				Technician		
17				Consumer		
18				3rd Party Provider	Operational Systems Provider	
19				Grid Provider		
20				3rd Party Provider	Infrastructure Service Provider	
21				Grid Provider		
22				3rd Party Provider	Business Service Provider	
23				Grid Provider		
24				3rd Party Provider	*Organizations*	
25				Grid Provider		
26	Active Resource	Grid Infrastructure Device, Micro Grid Infrastructure Device, Actuator, Sensor		**Grid Controller**	***Machine***	Component, Infrastructure Service, Business Service
27				**Provider System**		
28				**Grid Sub Controller**		
29				**Micro Grid Controller**		
30				**Micro Grid Sub Controller**		
31				**Other Device**		
32				Grid Controller	**Operational System**	
33				Provider System		
34				Grid Sub Controller		
35				Micro Grid Controller		
36				Micro Grid Sub Controller		
37				Other Device		
38				Meter		
39				Communication Hub		
40				Access Device		
41				Service Device		
42				Consumer Device		

Table 9.2.: *Smart Grid Pattern to SOA Stakeholder Pattern Relation Investigation Table*

systems provider, an infrastructure service provider, or a business service provider (see Table 9.2 rows 18, 20, 22, and 24 and column *Mapping*).

In case an element is written in italics, it means that this element is of another type as indicated by the row (see Table 9.1 row 15).

Example:
The machine of the SOA Stakeholder Pattern can be mapped to the elements grid (sub) controller, micro grid (sub) controller, provider system, or other device of the Smart Grid Pattern. But the elements grid (sub) controller, micro grid (sub) controller, provider system, and other device are of the type active resource in the Smart Grid Pattern as it has another point of view on the system-to-be (see Table 9.2 rows 2-7 and column *Mapping*).

A bold written element indicates that this element might be the reason for a refinement (see Table 9.1 row 2). Such a mapping can be bidirectional (see Table 9.1 row 10).

Example:
One might want to develop a SOA application in the first place. But later one is also interested to broaden the context to the smart grid domain as one of its central operational systems is part of a smart grid (see Table 9.2 row 32-42 and column *Mapping*). A bidirectional relation exists between, for example, machine and grid controller (see Table 9.2 row 2 and column *Mapping*). On the one hand, the refinement can start in a SOA context in which the machine realized as SOA is a grid controller. Thus, when the smart grid context is also important for our problem at hand, we use the Smart Grid Pattern jointly. On the other hand, the refinement can also start in the smart grid context and it turns out to be important that the grid controller is realized using a SOA.

A filled pattern relation investigation table serves different purposes. First, a pattern provider can use the table to derive the information whether the two patterns at hand are related and which type of relation applies. Second, in case a pattern consumer has already instantiated one pattern one can use the different tables related to the pattern at hand and check whether there is a reasonable way to extend the context or not (for more details see Section 9.2[16]). Third, in case a pattern consumer wants to combine two patterns the table can be used as guide for mapping elements of one instance to the other (see also Section 9.2[17]).

Table 9.3.: *Pattern Relation Reasoning Form*

Relation
Form n
Source Element The element which is under investigation
Target Element The element it is related to.
Relation Type ☐ Same The element and target element have the same semantic and can be used in both patterns by copying them over. ☐ Mapped The element can be used to instantiate a according target element. The mapping might not be one to one and the semantics of both elements might be different.
Refinement The source and / or target element is one possible cause for adding the context described by the context pattern which contains the opposite element. ☐ Source Refined ☐ Target Refined
Reliability ☐ Unreliable ☐ Reliable
Reasoning Some short description of the reasoning behind the relation
Source ☐ Thought Experiment ☐ Own Experience ☐ Expert Experience ☐ Documentation
Example(s) Some examples which give evidence to the existence of the relation

9.1.3. Forms for Analyzing and Documenting Relations between Context Pattern Elements

For analyzing and documenting relations between elements of two patterns at hand we propose to use so called *pattern relation reasoning forms*. Table 9.3 shows such a form, while Table 9.4 shows some example instances. The full table (Table B.26[18]) is part of the Appendix B.2[19]. Table 9.3 contains the *source element* which is related to a *target element*.

Example:
In Form 1 in Table 9.4 the grid controller is the source element and the machine is the target element.

Note that such a form is used to capture both navigation directions. Hence, a combination of two elements is reflected by only one form. A relation has a *relation type* which can be *same* or *mapping*.

Example:
In Form 1 in Table 9.4 the two elements are just mapped while in Form 7 in Table 9.4 they are the same.

Additionally, none (see Form 7 in Table 9.4), one (see Form 25 in Table 9.4) or both elements (see Form 1 in Table 9.4) might be the reason for a refinement of one context pattern using the other context pattern. This is the case if the same element instance can be the central element under consideration, for example the system-to-be, in both patterns. Hence, the context of this element instance is described by the combination of both context patterns. All this information is basically already reflected in the corresponding pattern relation investigation table, but it is also added here to have all information compiled in one place. The following information is then a reflection of the result of the (ongoing) mapping process. First the *reliability* is documented (Note that the examples do not contain any unreliable relations any longer as such relations are removed when the process of analyzing a relation between context patterns is finished), followed by a textual *reasoning* why the relation is reasonable (see all forms in Table 9.4). It is also documented which *sources* were used to establish the relation.

Example:
In Form 1 in Table 9.4 the relation between grid controller and the machine was directly derived from a discussion with an expert who also delivered an example.

It is also documented if there are *examples* which give evidence to the relation. Such a pattern reasoning form is useful for documenting the current state of an ongoing mapping process and is as such a target of change, but after the mapping is finished it is also useful for explaining a relation in the pattern relation investigation table to a pattern consumer.

[16]Page 171
[17]Page 171
[18]Page 450
[19]Page 446

Table 9.4.: *Pattern Relation Reasoning Forms for Smart Grid Pattern and SOA Stakeholder Pattern (Examples)*

Relation
Form 1
Source Element Grid Controller **Target Element** Machine **Relation Type** ☐ Same ☒ Mapped **Refinement** ☒ Source Refined ☒ Target Refined **Reliability** ☐ Unreliable ☒ Reliable **Reasoning** A serious problem of the IT infrastructure for a smart grid are the distributed nature and the interplay between different smart grid participants. The participants can change every time, for example a new 3rd party grid provider joins the market, so can the processes which are related to the smart grid. Hence, there are some efforts to tackle this problem by using a SOA for the smart grid IT infrastructure. As a result, a grid controller might be (partially) designed using a SOA and services. If the pattern user starts in the smart grid context and the grid controller is the central element of concern and therefore the system-to-be, he/she might want to explore its SOA context. Contrariwise, if someone who is developing a SOA application and it turns out that this application is later used to control a smart grid, he/she might gain additional useful insights by instantiating the Smart Grid Pattern. **Source** ☐ Thought Experiment ☐ Own Experience ☒ Expert Experience ☒ Documentation **Example(s)** One example is the Spectrum Power SOA offered by Siemens (2015 [348]). There are also some scientific efforts to combine a smart grid with SOA, for example the work of Yang, Lai, Chen, Liu, and Chu (2011 [394]).
Form 25
Source Element Other Device **Target Element** Operational System **Relation Type** ☐ Same ☒ Mapped **Refinement** ☐ Source Refined ☒ Target Refined **Reliability** ☐ Unreliable ☒ Reliable **Reasoning** In case another device is not the central system-to-be but is used as operational system for the system-to-be, the other device is mapped to an operational system in the SOA view. Contrariwise, an operational system, which is also directly integrated the smart grid itself might be mapped to another device. In case the operational system plays a central role in the SOA, it might be necessary to understand its smart grid context. Hence, it can be the reason for a refinement. Note that there might be operational systems in the SOA view which cannot be mapped to any Smart Grid Pattern element, as they are are not concerned with the smart grid at all. The other way around, another device might not be mapped at all, as it might not be related to the system-to-be build as SOA. **Source** ☒Thought Experiment ☐ Own Experience ☒ Expert Experience ☐ Documentation **Example(s)** –

Continued on next page

Table 9.4 – *continued from previous page*

Relation
Form 7
Source Element Legislator **Target Element** Legislator **Relation Type** ☒ Same ☐ Mapped **Refinement** ☐ Source Refined ☐ Target Refined **Reliability** ☐ Unreliable ☒ Reliable **Reasoning** Of course a legislator remains the same in a smart grid and SOA context. **Source** ☐ Thought Experiment ☒ Own Experience ☒ Expert Experience ☐ Documentation **Example(s)** –

9.1.4. Steps and Sources for Finding Relations between Context Patterns

While we introduced and discussed the pattern relation investigation table as means for relating two context patterns at hand in the previous section, we explain in this section the method a pattern provider can use to fill in such a table. The information for filling a pattern relation investigation table stems from three different sources:

Own Experience The first source are our experiences from using context patterns. As we have used such patterns extensively, we already came across situations where we combined context patterns (like in Beckers, Faßbender, and Schmidt (2012 [44])). Such known uses can be refined to the relations captured in the pattern relation investigation table.

Domain Experts The second source are experts of the different domains. They are often experts for one specific domain, but, from our experience, it is not unlikely that they were involved in cases where two or more domains were important.

Available Documentation The third source are reports or papers which document a certain combination of domains. Such documentation plays the role of known uses for a combination in our case. A documentation can be a more general description given in a white-paper, a product description, a project documentation, and so forth.

The process of filling a pattern relation investigation table and therefore relating the elements of two different context patterns starts with *selecting an exiting context pattern to be related* as shown in Figure 9.5. For the selected *existing context pattern* and the *new context pattern* we start with documenting the relations experienced by ourselves using a *pattern relation investigation table* and *relation reasoning forms*. Such relations are mostly supported by known uses of us. Hence, they are documented as reliable. Then, we conduct a *thought experiment* to find further possible relations. A possible relation is mostly supported by a crafted example. Such a relation is marked as unreliable in the according *relation reasoning form* and the *relation investigation table instance* is updated. After the thought experiment, we *search for existing available documentation* on combining the two domains at hand in general and for the unreliable relations in specific. For each documented relation found this way, we add a reliable relation, strengthen a reliable relation, or promote an unreliable relation. The reason that we do this thought experiment and the search for documentation is that in the next step we talk to domain experts and *document their feedback*. We experienced that it speeds up the process and raises the willingness of the experts to cooperate when the pattern relation investigation table is prefilled. The domain experts then give feedback about missing relations and relations they can confirm. For both cases they should give an example. This way, some of the relations marked as unreliable can be turned into reliable relations, or an already reliable relation is strengthened.

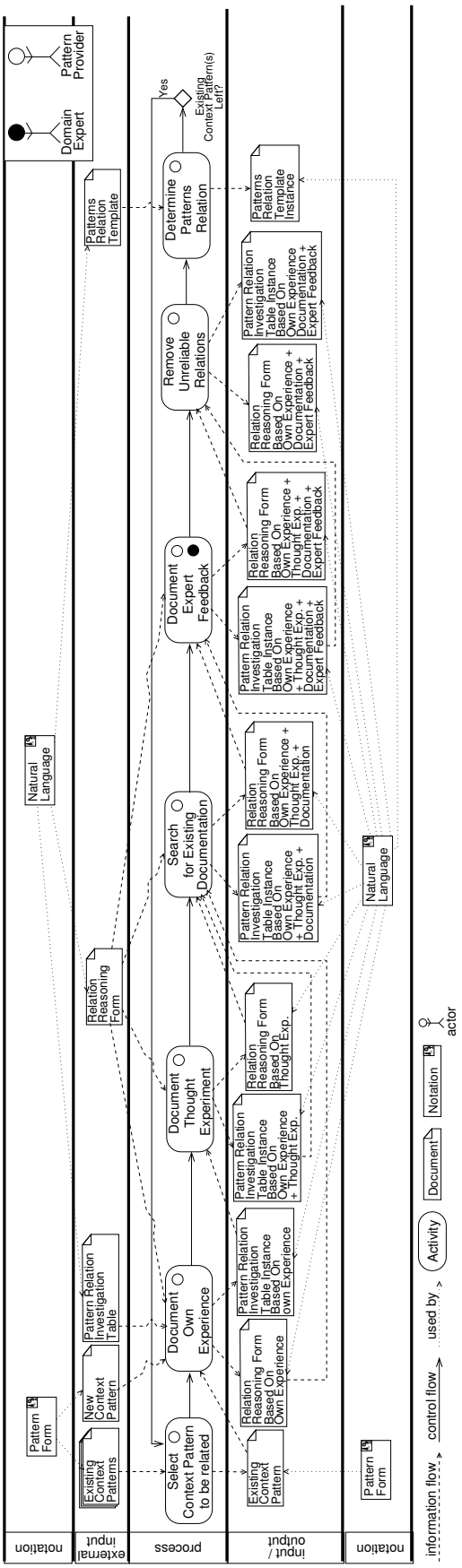


Figure 9.5.: *Patterns Relation Investigation Process*

Additionally, the documentation found should be discussed with domain experts to make sure that we do not is misinterpret the contained information. After conducting this step, we *remove* all remaining *unreliable relations* in the *pattern relation investigation table* at hand.

Last, we have to *determine the overall relation for the two patterns* at hand. The *specialization of* relation is defined by the way a context pattern was created. The *specialization of* relation only applies if one of the two patterns at hand was derived from the other and therefore covers a sub-domain of the parent pattern. If this is not the case, we take a look at the filled pattern investigation table. In case there is not a single element which is the same for both patterns or which can be mapped to another element of the opposite pattern, the two patterns are *unrelated*. In case there are elements which are the same or which can be mapped, but no element can be the reason for refinement, we have to check whether combinations of the elements, which are the same or which are mapped, can be used as reasonable input to one of the patterns. If this is the case, we have found an *input* relation. In case we have at least one element marked as potential reason for refinement, we have found a *can refine* relation. For patterns which are related by an *input* or a *can refine* relation, we have to check whether the two patterns at hand focus on different domains or if they describe the same domain but from different points of view. In the latter case we have also found a *used jointly* relation. Note that in most cases the *used jointly* relation is known beforehand, because patterns which can be used jointly are in most cases also derived and described along with each other.

9.1.5. Results of the Relation Mining

In this section, we summarize the results of the relation mining in a brief form. The detailed results can be found in Appendix B.2[20]. We describe each relation between two context pattern using a so called *patterns relation template* (see Table 9.5 for an example. The rest if the instances can be found in Appendix B.2[21]) that states first the *Direction* of the relation, second the *Relation Type*, third the *Reasoning* why the relationship holds, and forth a reference to the *details* of the relation. Relations between context patterns can consist of several (sub-)relations. The relation type and direction are derived from relation investigation tables as already described in Section 9.1.4. The template instance only contains a compact reasoning and explanation of the relation. The found relations are enumerated in the following:

SOA Stakeholder Pattern ↔ Smart Grid Pattern The SOA Stakeholder Pattern and the Smart Grid Pattern *can refine* each other (see Table B.15[22]).

Cloud System Analysis Pattern ↔ Smart Grid Pattern The Cloud System Analysis Pattern and the Smart Grid Pattern *can refine* each other (see Table B.16[23]).

SOA Stakeholder Pattern ↔ Cloud System Analysis Pattern The SOA Stakeholder Pattern and the Cloud System Analysis Pattern *can refine* each other (see Table 9.5).

Cloud System Analysis Pattern ↔ Peer to Peer Pattern The Peer to Peer Pattern *can refine* the Cloud System Analysis Pattern (see Tab. B.17[24]).

Smart Grid Pattern ↔ Peer to Peer Pattern The Peer to Peer Pattern can refine the Smart Grid Pattern (see Tab. B.18[25]).

SOA Stakeholder Pattern ↔ SOA Layer Pattern The SOA Stakeholder Pattern and the SOA Layer Pattern *can refine* each other and are meant to be *used jointly*. (see Table. B.19).

SOA Stakeholder Pattern ↔ Peer to Peer Pattern The SOA Stakeholder Pattern *can be refined* by a Peer to Peer Pattern instance (see Table. B.21).

[20] Page 446
[21] Page 446
[22] Page 446
[23] Page 446
[24] Page 446
[25] Page 447

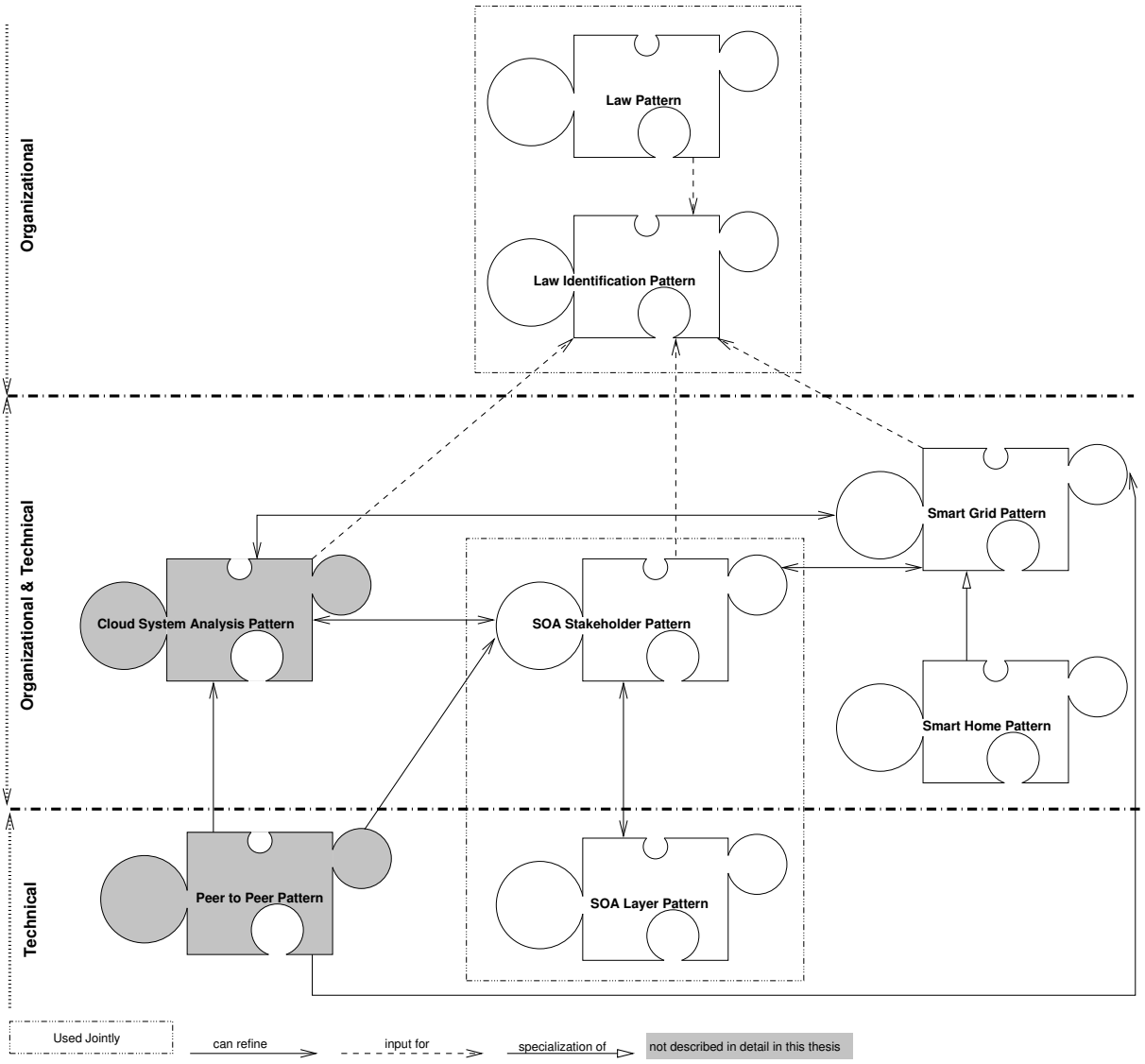


Figure 9.6.: *Relations between context patterns*

SOA Stakeholder Pattern ↔ Law Identification Pattern Our SOA Stakeholder Pattern can be *input* for the Law Identification Patterns to identify relevant laws for SOA scenarios.(see Table. B.20)

Cloud System Analysis Pattern ↔ Law Identification Pattern Our Cloud System Analysis Pattern can be *input* for the Law Identification Pattern.(see Table. B.22) An example of how to use this relation is shown in Beckers, Faßbender, and Schmidt (2012 [44]).

Law Pattern ↔ Law Identification Pattern Our Law Pattern can be *input* for the Law Identification Patterns as it provides the legal terms needed classifying the technical terms. Both patterns are also meant to be *used jointly* (see Table. B.23)

Smart Grid Pattern ↔ Law Identification Pattern Our Smart Grid Pattern can be *input* for the Law Identification Pattern.(see Table. B.24)

The found relations between context patterns can be also visualized to get a quick overview as shown in Figure 9.6. In general, we have three groups of context patterns. Context patterns which only focus on the technical context, context patterns that only focus on organizational aspects, and context patterns which combine those two views (the detailed discussion on the

Table 9.5.: *Pattern Relation SOA Stakeholder Pattern to Cloud System Analysis Pattern*

Direction	SOA Stakeholder Pattern to Cloud System Analysis Pattern, Cloud System Analysis Pattern to SOA Stakeholder Pattern
Relation Type	can refine
Reasoning	The services deployed in a cloud can be created or composed in a SOA. Hence, the information in the SOA Stakeholder Pattern can be seen as a refinement of the services in the Cloud System Analysis Pattern. In addition, the stakeholders involved in the creation and maintenance of the service can be cloud stakeholders. But it is also possible that the Cloud System Analysis Pattern is used to elicit more information about stakeholders of a SOA Stakeholder Pattern or the deployment of the whole system-to-be.
Details	Table B.27 (Page 464)

different groups can be found in Chapter 7[26]). The three groups are indicated in Figure 9.6 as layers separated by dot-slashed lines.

The relations are shown using directed arrows. A solid arrow with an open head indicates a *can refine* relation, a solid arrow with a closed head visualizes a *specialization of* relation, while a dashed arrow indicates an *input* relation. Some of the patterns are *used jointly* which means that those patterns are usually used together and closely related. For example, the SOA Stakeholder Pattern contains all layers and elements of the SOA Layer Pattern. The SOA Layer Pattern only adds the technical relations between the elements, while the SOA Stakeholder Pattern adds the environment and stakeholders. The *can refine* relation has a particular effect when it occurs between a pattern which can be *used jointly* with other context patterns and a context pattern which cannot be used jointly with the first pattern. Namely, the relation always occurs for all context patterns which can be used jointly, but this is not explicitly shown. For example, the SOA Stakeholder Pattern can refine the Cloud System Analysis Pattern, so can the SOA Layer Pattern. But in a normal case, both SOA patterns will be used jointly to refine a Cloud System Analysis Pattern.

9.1.6. Lessons learned

While the process description given in Section 9.1.4 sounds like a straightforward process, it is not in reality. Some possible relations do not come up during a thought experiment session, but at another point in time. The opportunities to talk to domain experts often drop in by accident. Cross-domain documents are found while searching for other content. This leads to a more or less unstructured, iterative process. But one has to make sure that all four sources of information are considered and reflected in the final pattern relation investigation table thoroughly. The ideal scenario for the process mentioned above would be a sequence of workshops following the process. But the given constraints in time and budget often hinder such an ideal scenario.

Another lesson learned is about the effort for establishing relations. At first, it seems that for adding one additional pattern one has to make numberOfOldPatterns comparisons. This turns out to be the upper bound never reached. The reasons are manifold. First, a new pattern has only to be related to patterns of the same group and adjacent group (for example, an organizational pattern has to be related to other organizational patterns and to organizational & technical context patterns (adjacent group)). We never came across a situation where we were able to relate a technical-only context pattern with an organizational-only context pattern. Second, if already existing context patterns are meant to be used jointly or one pattern is a specialization of another, it is sufficient to relate the new pattern to one of them. In this case the relations found for the general pattern also apply for the specialized pattern as explained in Section 9.1.5. Third, there are domains which exclude each other at first sight. Nevertheless, the

[26] Page 113

effort for relating context patterns is noticeable. It is possible to learn from already established relations between existing context patterns, for example, in means of inspiration for the thought experiments or already found documentation. This way, the effort can be lowered once again. One should also keep in mind that the effort has only to be spent once by the context pattern provider, while context pattern consumers benefit each time they search for a combination of context patterns which is sufficient for their problem at hand. And as the latter case occurs much more often than the context pattern relation case, the effort spend for relating patterns pays off.

The next lesson learned is to wait until a new context pattern is really stable. Every change of the pattern, when, for example, a new element is added, an element is removed, or the semantic meaning of an element has to be changed, leads inevitably to a rerun of the relation establishing process. Finally, we acknowledge that we only have one evaluation of our process (the application to the patterns prsented in this work), so far. In the future, we are planning to conduct further evaluations of the process and share further lessons learned.

9.2. Navigating and Combining Context Patterns

In this section we show how someone who is actually in search for a way to describe the complete relevant context for a problem at hand can use the relations and the relation investigation tables for solving his / her problem. Hence, this section is not meant for pattern providers like Section 9.1.4, but for pattern consumers.

Example:
We use our running example Media Market as described in Chapter 5[27] to exemplify the method.

The first step of using context patterns is to *select the initial, most important one* as shown in Figure 9.7. Therefore, we look at the *relation overview* as depicted in Figure 9.6 to get a quick overview. For those context patterns which are promising for describing the context in question, we use the *catalog* (Chapter 8[28]) to get a quick impression of the pattern candidates by reading the summary. To finish the first step, we have to *select the initial context pattern*. We suggest to take a context pattern which combines an organizational and technical view first. Such a pattern provides a comprehensive overview of a domain at hand.

Example:
For our media market, the central context is the SOA domain. Hence, after looking at the overview, we select the two SOA related context patterns as candidates. After reading the summary, we select the SOA stakeholder pattern as it describes a broader context. The technical view the SOA Layer Pattern describes is not of central relevance in the first place.

Next, we *instantiate the selected initial context pattern* using the solution as described in the according pattern form for the *initial context pattern* in Chapter 8[29].

Example:
The result of the instantiation of the SOA Stakeholder pattern is shown in Chapter 10[30].

[27] Page 75
[28] Page 131
[29] Page 131

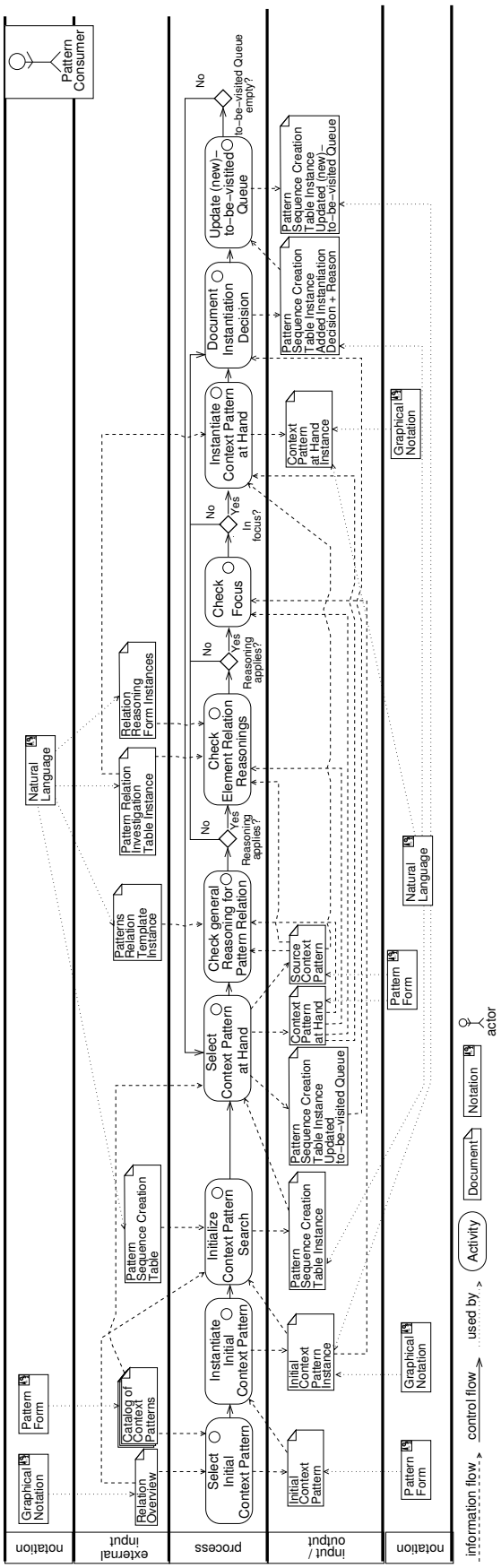


Figure 9.7.: *Patterns Combination Process*

Step	Pattern at Hand	Source Pattern	To-Be-Visited Queue	Visited Queue	Relations for Pattern at Hand	New To-Be-Visited Queue	Instantiated?	Reasoning
Init	pat1				**pat2**, *pat3*, pat4	**(pat1, pat2)**, (pat1, pat4), (pat1, *pat3*)	✔	Most important
1	pat2	pat1	(pat1, pat4), *(pat1, pat3)*	**(pat1, pat2)**	**pat1**, pat4	(pat1, pat4), *(pat1, pat3)*, (pat2, pat4)	✔	describes technical context in detail
2	pat4	pat1	*(pat1, pat3)*, (pat2, pat4)	**(pat1, pat2), (pat1, pat4)**	pat1, pat4	*(pat1, pat3)*, (pat2, pat4)	✖	No input possible
...	...	...	...	...	...	...	...	...

can refine, **jointly used, specialization of**, *input*

Table 9.6.: *Pattern Sequence Creation Table*

Now, we *initialize a breadth-first search*, because the closer one context pattern is to the initial context pattern in terms of refinement steps, the more likely it is that it might be also relevant. For keeping track of the search, we propose to use so called *pattern sequence creation tables*. Table 9.6 shows such a table. The table is initialized by using the *initially selected context pattern*. Therefore, the initial context pattern is the first *pattern at hand* (Table 9.6 Step initial, column Pattern at Hand). For the pattern at hand we collect the according *relations* to other context patterns (Table 9.6 Step initial, column Relations for Pattern at Hand). Of course, the initial pattern is instantiated (Table 9.6 Step initial, column Instantiated?) and some reasoning why the pattern at hand is instantiated or not is provided (Table 9.6 Step initial, column Reasoning). We add the found relations for the pattern at hand to a *new to-be-visited queue* (Table 9.6 Step initial, column New To-Be-Visited Queue). We suggest to order the relations first before putting them into the queue. The order should be *specialization of*, *jointly used*, *can refine*, and *input*. The order reflects how close the different relation types relate two patterns. Hence, it seems to be reasonable to look at those patterns which are very closely related, because, for example, one pattern is a specialization of the other. Note that a pattern or relation given in **bold** visualizes a *jointly used* or *specialization of* relation, *input* relations are given in *italics*, and normal font visualizes a *can refine* relation.

Example:
Table 9.7 shows the pattern sequence creation table for our example. The row init shows the initialization of the table. As we have chosen the SOA Stakeholder Pattern for the initial pattern, we get the according relations to the SOA Layer Pattern, the Cloud System Analysis Pattern, the Peer To Peer Pattern, and the Smart Grid Pattern. Of course, our initial pattern is instantiated. Hence, we add these relations the new to-be-visited queue.

Next, we start the search by *selecting* the first element from the new to-be-visited queue as *pattern at hand* (Table 9.6 Step 2, column Pattern at Hand) and we also keep in mind the *source pattern* of the relation (Table 9.6 Step 2, column Source Pattern). The reduced new to-be-visited queue is stored as *to-be-visited queue* (Table 9.6 Step 2, column To-Be-Visited Queue). The relation we just picked is added to the visited queue (Table 9.6 Step 2, column Visited Queue).

Example:
We pick the relation between SOA Stakeholder Pattern and SOA Layer Pattern from the new to-be-visited queue. (Table 9.7, Step 1). Hence, our pattern at hand is the SOA Layer Pattern, while the source pattern is the SOA Stakeholder Pattern.

Then we have to decide whether the pattern at hand can be used to extend the context of the

[30] Page 181

Step	Pattern at Hand	Source Pattern	To-Be-Visited Queue	Visited Queue	Relations for Pattern at Hand	New To-Be-Visited Queue	Instantiated?	Reasoning
Init	SOA Stakeholder Pattern				**SOA Layer Pattern**, Cloud System Analysis Pattern, Smart Grid Pattern, Peer to Peer Pattern, *Law Identification Pattern*	**(SOA Stakeholder Pattern, SOA Layer Pattern)**, (SOA Stakeholder Pattern, Cloud System Analysis Pattern), (SOA Stakeholder Pattern, Smart Grid Pattern), (SOA Stakeholder Pattern, Peer to Peer Pattern), *(SOA Stakeholder Pattern, Law Identification Pattern)*	✓	The media market shall be a SOA application.
1	SOA Layer Pattern	SOA Stakeholder Pattern	(SOA Stakeholder Pattern, Cloud System Analysis Pattern), (SOA Stakeholder Pattern, Smart Grid Pattern), (SOA Stakeholder Pattern, Peer to Peer Pattern), *(SOA Stakeholder Pattern, Law Identification Pattern)*	**(SOA Stakeholder Pattern, SOA Layer Pattern)**	**SOA Stakeholder Pattern**, Cloud System Analysis Pattern, Smart Grid Pattern, Peer to Peer Pattern, *Law Identification Pattern*	(SOA Stakeholder Pattern, Cloud System Analysis Pattern), (SOA Stakeholder Pattern, Peer to Peer Pattern), *(SOA Stakeholder Pattern, Law Identification Pattern)*, (SOA Layer Pattern, Cloud System Analysis Pattern), (SOA Layer Pattern, Smart Grid Pattern), (SOA Layer Pattern, Peer to Peer Pattern)	✓	Describes technical SOA context in detail
2	Cloud System Analysis Pattern	SOA Stakeholder Pattern	(SOA Stakeholder Pattern, Smart Grid Pattern), (SOA Stakeholder Pattern, Peer to Peer Pattern), *(SOA Stakeholder Pattern, Law Identification Pattern)*, (SOA Layer Pattern, Cloud System Analysis Pattern), (SOA Layer Pattern, Smart Grid Pattern), (SOA Layer Pattern, Peer to Peer Pattern), *(SOA Layer Pattern, Law Identification Pattern)*	**(SOA Stakeholder Pattern, SOA Layer Pattern)**, (SOA Stakeholder Pattern, Cloud System Analysis Pattern)	SOA Stakeholder Pattern, SOA Layer Pattern, Smart Grid Pattern, Peer to Peer Pattern, *Law Identification Pattern*	(SOA Stakeholder Pattern, Smart Grid Pattern), (SOA Stakeholder Pattern, Peer to Peer Pattern), *(SOA Stakeholder Pattern, Law Identification Pattern)*, (SOA Layer Pattern, Cloud System Analysis Pattern), (SOA Layer Pattern, Smart Grid Pattern), (SOA Layer Pattern, Peer to Peer Pattern), *(Cloud System Analysis Pattern, Law Identification Pattern)*	✓	Our media market service is supposed to run in a cloud, as we have strong requirements regarding scalability and avaiability.
3	Smart Grid Pattern	SOA Stakeholder Pattern	(SOA Stakeholder Pattern, Cloud System Analysis Pattern), (SOA Stakeholder Pattern, Smart Grid Pattern), (SOA Stakeholder Pattern, Peer to Peer Pattern), *(SOA Stakeholder Pattern, Law Identification Pattern)*	**(SOA Stakeholder Pattern, SOA Layer Pattern)**, (SOA Stakeholder Pattern, Cloud System Analysis Pattern), (SOA Stakeholder Pattern, Smart Grid Pattern)	SOA Stakeholder Pattern, SOA Layer Pattern, Cloud System Analysis Pattern, Peer to Peer Pattern, *Law Identification Pattern*	(SOA Stakeholder Pattern, Peer to Peer Pattern), (SOA Stakeholder Pattern, *Law Identification Pattern*), (SOA Layer Pattern, Smart Grid Pattern), (SOA Layer Pattern, Peer to Peer Pattern), (Cloud System Analysis Pattern, Smart Grid Pattern), (Cloud System Analysis Pattern, Peer to Peer Pattern), *(Cloud System Analysis Pattern, Law Identification Pattern)*	✖	Our media market is not related to the smart grid at all, are the operational systems, backend services, or infrastructure services it relies an.
4	Peer to Peer Pattern	SOA Stakeholder Pattern	(SOA Stakeholder Pattern, Law Identification Pattern), (SOA Layer Pattern, Smart Grid Pattern), (SOA Layer Pattern, Peer to Peer Pattern), *(SOA Layer Pattern, Law Identification Pattern)*, (Cloud System Analysis Pattern, Smart Grid Pattern), (Cloud System Analysis Pattern, Peer to Peer Pattern), *(Cloud System Analysis Pattern, Law Identification Pattern)*	**(SOA Stakeholder Pattern, SOA Layer Pattern)**, (SOA Stakeholder Pattern, Cloud System Analysis Pattern), (SOA Stakeholder Pattern, Smart Grid Pattern), (SOA Stakeholder Pattern, Peer to Peer Pattern)	SOA Stakeholder Pattern, SOA Layer Pattern, Cloud System Analysis Pattern, Smart Grid Pattern	*(SOA Stakeholder Pattern, Law Identification Pattern)*, (SOA Layer Pattern, Smart Grid Pattern), (SOA Layer Pattern, Peer to Peer Pattern), (Cloud System Analysis Pattern, Smart Grid Pattern), (Cloud System Analysis Pattern, Peer to Peer Pattern), *(Cloud System Analysis Pattern, Law Identification Pattern)*	✖	There is no reason to use peer to peer technologies for our media market.
5	Law Identification Pattern	SOA Stakeholder Pattern	(SOA Layer Pattern, Smart Grid Pattern), (SOA Layer Pattern, Peer to Peer Pattern), (Cloud System Analysis Pattern, Smart Grid Pattern), (Cloud System Analysis Pattern, Peer to Peer Pattern), *(Cloud System Analysis Pattern, Law Identification Pattern)*.	**(SOA Stakeholder Pattern, SOA Layer Pattern)**, (SOA Stakeholder Pattern, Cloud System Analysis Pattern), (SOA Stakeholder Pattern, Smart Grid Pattern), (SOA Stakeholder Pattern, Peer to Peer Pattern), *(SOA Stakeholder Pattern, Law Identification Pattern)*.	**Law Pattern**	(SOA Layer Pattern, Smart Grid Pattern), (SOA Layer Pattern, Peer to Peer Pattern), (Cloud System Analysis Pattern, Smart Grid Pattern), (Cloud System Analysis Pattern, Peer to Peer Pattern), *(Cloud System Analysis Pattern, Law Identification Pattern)*, **(Law Identification Pattern, Law Pattern)**	✓	The realization of the media market is highly regulated by laws. Hence, we are interested in getting the legal context.
6	Smart Grid Pattern	SOA Layer Pattern	(SOA Layer Pattern, Peer to Peer Pattern), (Cloud System Analysis Pattern, Smart Grid Pattern), (Cloud System Analysis Pattern, Peer to Peer Pattern), *(Cloud System Analysis Pattern, Law Identification Pattern)*, ***(Law Identification Pattern, Law Pattern)***	**(SOA Stakeholder Pattern, SOA Layer Pattern)**, (SOA Stakeholder Pattern, Cloud System Analysis Pattern), (SOA Stakeholder Pattern, Smart Grid Pattern), (SOA Stakeholder Pattern, Peer to Peer Pattern), *(SOA Stakeholder Pattern, Law Identification Pattern)*, (SOA Layer Pattern, Smart Grid Pattern)	SOA Stakeholder Pattern, SOA Layer Pattern, Cloud System Analysis Pattern, Smart Grid Pattern	*(SOA Stakeholder Pattern, Law Identification Pattern)*, (SOA Layer Pattern, Peer to Peer Pattern), (Cloud System Analysis Pattern, Smart Grid Pattern), (Cloud System Analysis Pattern, Peer to Peer Pattern), *(Cloud System Analysis Pattern, Law Identification Pattern)*, **(Law Identification Pattern, Law Pattern)**	✖	See Step 3
7	Smart Grid Pattern	SOA Layer Pattern	(SOA Layer Pattern, Peer to Peer Pattern), (Cloud System Analysis Pattern, Smart Grid Pattern), (Cloud System Analysis Pattern, Peer to Peer Pattern), *(Cloud System Analysis Pattern, Law Identification Pattern)*, ***(Law Identification Pattern, Law Pattern)***	**(SOA Stakeholder Pattern, SOA Layer Pattern)**, (SOA Stakeholder Pattern, Cloud System Analysis Pattern), (SOA Stakeholder Pattern, Smart Grid Pattern), (SOA Stakeholder Pattern, Peer to Peer Pattern), *(SOA Stakeholder Pattern, Law Identification Pattern)*, (SOA Layer Pattern, Smart Grid Pattern)	SOA Stakeholder Pattern, **Cloud System Analysis Pattern**, SOA Layer Pattern, Peer to Peer Pattern, *Law Identification Pattern*	(SOA Layer Pattern, Peer to Peer Pattern), (Cloud System Analysis Pattern, Smart Grid Pattern), (Cloud System Analysis Pattern, Peer to Peer Pattern), *(Cloud System Analysis Pattern, Law Identification Pattern)*, **(Law Identification Pattern, Law Pattern)**	✖	See Step 3
8	Peer to Peer Pattern	SOA Layer Pattern	(Cloud System Analysis Pattern, Smart Grid Pattern), (Cloud System Analysis Pattern, Peer to Peer Pattern), *(Cloud System Analysis Pattern, Law Identification Pattern)*, ***(Law Identification Pattern, Law Pattern)***	**(SOA Stakeholder Pattern, SOA Layer Pattern)**, (SOA Stakeholder Pattern, Cloud System Analysis Pattern), (SOA Stakeholder Pattern, Smart Grid Pattern), (SOA Stakeholder Pattern, Peer to Peer Pattern), *(SOA Stakeholder Pattern, Law Identification Pattern)*, (SOA Layer Pattern)	SOA Stakeholder Pattern, SOA Layer Pattern, Cloud System Analysis Pattern, Smart Grid Pattern	(Cloud System Analysis Pattern, Smart Grid Pattern), (Cloud System Analysis Pattern, Peer to Peer Pattern), *(Cloud System Analysis Pattern, Law Identification Pattern)*, **(Law Identification Pattern, Law Pattern)**	✖	See Step 4
9	Smart Grid Pattern	Cloud System Analysis Pattern	(Cloud System Analysis Pattern, Peer to Peer Pattern), *(Cloud System Analysis Pattern, Law Identification Pattern)*, ***(Law Identification Pattern, Law Pattern)***	**(SOA Stakeholder Pattern, SOA Layer Pattern)**, (SOA Stakeholder Pattern, Cloud System Analysis Pattern), (SOA Stakeholder Pattern, Smart Grid Pattern), (SOA Stakeholder Pattern, Peer to Peer Pattern), *(SOA Stakeholder Pattern, Law Identification Pattern)*, (SOA Layer Pattern, Smart Grid Pattern), (Cloud System Analysis Pattern, Smart Grid Pattern)	SOA Stakeholder Pattern, **SOA Layer Pattern**, **Cloud System Analysis Pattern**, **Peer to Peer Pattern**, *Law Identification Pattern*	(Cloud System Analysis Pattern, Peer to Peer Pattern), *(Cloud System Analysis Pattern, Law Identification Pattern)*, **(Law Identification Pattern, Law Pattern)**	✖	The cloud system analysis pattern does not imply any connection to the smart grid domain by default. And the original SOA context does not contain any smart grid relation either.
10	Peer to Peer Pattern	Cloud System Analysis Pattern	***(Cloud System Analysis Pattern, Law Identification Pattern)***, **(Law Identification Pattern, Law Pattern)**	**(SOA Stakeholder Pattern, SOA Layer Pattern)**, (SOA Stakeholder Pattern, Cloud System Analysis Pattern), (SOA Stakeholder Pattern, Smart Grid Pattern), (SOA Stakeholder Pattern, Peer to Peer Pattern), *(SOA Stakeholder Pattern, Law Identification Pattern)*, (SOA Layer Pattern, Peer to Peer Pattern),(Cloud System Analysis Pattern, Peer to Peer Pattern)	SOA Stakeholder Pattern, SOA Layer Pattern, Cloud System Analysis Pattern, Smart Grid Pattern	*(Cloud System Analysis Pattern, Law Identification Pattern)*, **(Law Identification Pattern, Law Pattern)**	✖	The Peer to Peer pattern is relevant for the Cloud System Analysis Pattern Instance. Parts of the cloud, on which our media market service will be deployed, are using peer to peer technologies. But these technical details are up to the cloud provider to deal with and not of our concern.
11	Law Identification Pattern	Cloud System Analysis Pattern	(Law Identification Pattern, Law Pattern)	**(SOA Stakeholder Pattern, SOA Layer Pattern)**, (SOA Stakeholder Pattern, Cloud System Analysis Pattern), (SOA Stakeholder Pattern, Smart Grid Pattern), (SOA Stakeholder Pattern, Peer to Peer Pattern), *(SOA Stakeholder Pattern, Law Identification Pattern)*, (SOA Layer Pattern, Cloud System Analysis Pattern), (SOA Layer Pattern, Smart Grid Pattern), *(Cloud System Analysis Pattern, Law Identification Pattern)*	**Law Pattern**	(Law Identification Pattern, Law Pattern)	✓	The cloud deployment of our media market might add some additional legal considerations.
12	Law Pattern	Law Identification Pattern		**(SOA Stakeholder Pattern, SOA Layer Pattern)**, (SOA Stakeholder Pattern, Cloud System Analysis Pattern), (SOA Stakeholder Pattern, Smart Grid Pattern), (SOA Stakeholder Pattern, Peer to Peer Pattern), *(SOA Stakeholder Pattern, Law Identification Pattern)*, (SOA Layer Pattern, Cloud System Analysis Pattern), (SOA Layer Pattern, Smart Grid Pattern), (SOA Layer Pattern, Peer to Peer Pattern), (Cloud System Analysis Pattern, Smart Grid Pattern), (Cloud System Analysis Pattern, Peer to Peer Pattern), *(Cloud System Analysis Pattern, Law Identification Pattern)*, **(Law Identification Pattern, Law Pattern)**	Law Identification Pattern	✓	Of course we want to use the Law Pattern to prepare laws for the matching.

can refine, **jointly used, specialization of**, *input*

Table 9.7.: *Pattern Sequence Creation Table Example*

source pattern. Therefore, we *check the brief reasoning for a relation* as provided in Section 9.1.5 by the different *patterns relation template* instances.

Example:
In case of step 1, the brief reasoning given in Table B.19[31] implies a reasonable usage of the SOA Layer Pattern, because we are not only interested in the organizational context and the relation between the different stakeholders to the technical elements of the SOA-to-be, but also in the detailed relations between technical elements. In contrast, in step 3 we can already derive from the brief reasoning given in Table B.15[32] for the relation that the Smart Grid Pattern does not need to be considered, as our Media Market does not contain any technical element which could also be part of a smart grid.

If the reasoning implies the possibility of a context extension, we have to *check the element relations* between the elements of the two patterns at hand in detail. To do so, we take a look at the according *pattern relation investigation table* (see Section 9.1.2[33] and Appendix B[34]). We have to check all elements of the source pattern which are given in bold indicating a refinement relation. For each of these elements we read the according *reasoning form* (see Section 9.1.2[35] and Section 9.1.4[36]). If the reasoning for a refinement also applies for the problem at hand and the according instances for the element at hand, we can instantiate the context pattern. If it does not apply, we check the next element given in bold. In the end, if we do not find any reasoning for refinement, we do not instantiate the context pattern at hand. If we have found some reasonable reasoning, we *check the focus*. Hence, we check whether we would detach from the original problem by instantiating the pattern at hand which means that the context as described by the pattern at hand is not of relevance for the original problem at hand. This can happen as there might be several refinement steps between the initial context pattern chosen and the current context pattern. Hence, a pattern reached might be a reasonable refinement when only looking at its direct predecessor, but as it might be reached after several refinement steps it might be out of focus when reconsidering the problem for which the context has to be elicited. In case the newly added information by instantiating the context pattern at hand is valuable and still of relevance for the problem at hand, we decide to *instantiate the pattern at hand*. If there are any specializations of the pattern at hand which are more suitable for our problem, we replace the pattern at hand with the specialized pattern.

Example:
In case of step 1 (Table 9.7 first column), the pattern relation investigation table (See Table B.30[37]), shows that we can refine the machine constituent using the SOA Layer Pattern. In case of step 4, we do not find any element, which might imply a refinement. The reason is that we are not planning to use any peer to peer technology to implement any of our Media Market services. In case of step 10, we find different elements, such as the hypervisor or database, which imply a refinement of the Cloud System Analysis Pattern instance using the Peer to Peer Pattern. But these elements are not of concern for our media market as they are of relevance for the cloud provider, but we are the cloud customer. For us only the service level agreement

[31] Page 447
[32] Page 446
[33] Page 159
[34] Page 395
[35] Page 159
[36] Page 166

(SLA) with the cloud provider is of relevance and it is not of relevance which technical means the provider uses to guarantee the fulfillment of the SLA. Hence, we do not instantiate the Peer to Peer Pattern.

The instantiation of the *pattern at hand* based on the *source pattern* is also guided by the *pattern relation investigation table*. Those elements of the pattern at hand which are exclusive to the pattern at hand have to be elicited from scratch. Those elements which are the same have to be copied from the instance of the source pattern to the instance of the pattern at hand. And for those elements which have a mapping defined by the pattern relation investigation table one can use the according reasoning form to decide whether the mapping applies for the elements of the source pattern instance.

Example:
The result of the instantiation of the different patterns and how they are combined with each other are shown in Chapter 10[38].

The *result of the overall decision* if the pattern at hand is instantiated or not *is documented* in the *pattern sequence creation table* (Table 9.6 Step 2 and Step 3, column Instantiated? and Reasoning). In case we decide to instantiate the pattern at hand, we add the relations to other context patterns for the pattern at hand to the new to-be-visited queue (Table 9.6 Step 2, column new To-Be-Visited Queue). Afterward, we remove those relations which were already visited (Table 9.6 Step 2 and Step 3, column new Visited Queue) or those relations for which the target pattern is already instantiated and not related by an input relation to the source pattern (Table 9.6 column Pattern at Hand, column Instantiated). In case the pattern at hand is not instantiated, we copy the to-be-visited queue over to the new to-be-visited queue (Table 9.6 Step 3, column new To-Be-Visited Queue). Unfortunately, we cannot remove all relations considering the not instantiated pattern at hand, because we might not find a reason for refining the current relation, but having another source pattern for the pattern at hand, we might find such a reasonable refining.

Then, we select the next relation from the new to-be-visited queue and proceed as described. The process is stopped when the new to-be-visited queue is empty.

Example:
The complete result of applying the described process is shown in Table 9.7. The resulting sequence of patterns as depicted in Figure 9.8 is *SOA Stakeholder Pattern, SOA Layer Pattern, Cloud System Analysis Pattern, Law Identification Pattern, Law Pattern.*

Note that the process as described in this section is rather difficult to apply by hand and the effort is considerable. But it is also easily implemented by a tool which guides the pattern user through the process, and maintains the pattern sequence creation table automatically. Moreover, form our experience this process outperforms a process in which each pattern is analyzed in isolation for its applicability. The reason is that the pattern relation investigation tables and the according reasoning forms contain much information about possible relations between two patterns. In most cases, the pattern users does not know this information by themselves. In

[37]Page 466
[38]Page 181

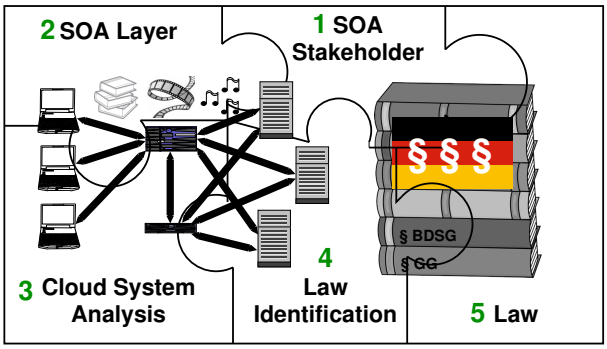


Figure 9.8.: *Patterns Sequence for the Media Market*

consequence, looking at a pattern alone to judge its applicability bears the risk of rejecting it, because the reason for using it does not directly comes to the mind of the pattern user. Hence, using our proposed process assures that pattern users consider all possible relations and reasons for combining context patterns in a structured way.

9.3. A Pattern Language for Context Patterns

We describe in the following how our work done so far on context patterns fits into the required elements of a pattern language and discuss how close we are to having a pattern language for context patterns. This discussion is based on the insights provided in Section 6.5[39].

Patterns (Vocabulary) We analyzed in Section 7.3.1[40] which elements and concepts we used in the context patterns presented in different works of ours (see Chapter 8[41] and according Appendix B.1[42] for the catalog). Further, we also described the relations between the identified elements and concepts in a meta model in Section 7.3.1[43] and we showed *how* to describe a context pattern using the meta model in Section 7.4[44].

Each of our context patterns addresses a particular problem and has a method that states how to solve this problem. The pattern form which reflects this information was introduced in Section 8.1[45]. We explicitly state the problem and forces for the problem. Describing the solution in a method eases the application of the solution for engineers.

We claim that our meta model, the pattern form and the pattern catalog contain the vocabulary for our pattern language and published the claim in (Beckers, Faßbender, and Heisel, 2013 [47]; Beckers, Faßbender, and Heisel, 2014 [49]).

Connections between Patterns (Syntax) We analyzed the relations between our context patterns and presented the results in Section 9.1.1[46]. Context patterns *can refine* each other or one pattern is a *specialization of* another pattern. Additionally, the domain knowledge elicited and stored in one context pattern can be used by another pattern as *input*. Some of the context patterns are meant to be used *jointly*. The relation investigation tables as

[39] Page 102
[40] Page 121
[41] Page 131
[42] Page 396
[43] Page 121
[44] Page 126
[45] Page 132
[46] Page 152

introduced in Section 9.1.2[47] were instantiated for our existing context patterns and abstracted to the general relations (See Section 9.1.5[48]) between the context patterns which form the syntax. We claim that this way we formed the syntax and published the claim in (Beckers, Faßbender, and Heisel, 2014 [50]).

Pattern Sequences (Grammar) We are showing all relations between our context patterns in Figure 9.6[49]. These relations can be used to combine different context patterns to sequences as shown in Section 9.2[50]. Hence, Figure 9.6[51] implicitly describes all possible sequences, which is similar to the related works by Eloranta et al. (2014 [126]), Buschmann et al. (1996 [91]), Gamma et al. (1994 [154]), Schumacher et al. (2006 [342]), and Schümmer and Lukosch (2007 [343]). An initial idea how to combine patterns is already provided by the pattern relation investigation table as discussed in Section 9.2[52]. But for recurring combinations it might be reasonable to have a new method for this particular combination. We showed how this can work in Beckers, Faßbender, and Schmidt (2012 [44]) and combined the Cloud System Analysis Pattern with the Law Pattern adding a new method. Section 9.2[53] also shows how to use the Figure 9.6[54] and the pattern relation investigation tables to actually find a proper sequence for a problem at hand. The figure shows the relations between context patterns. Hence, a sequence contains context patterns that are used jointly, where one pattern refines another or one pattern is input for another context pattern. The relation investigation tables as introduced in Section 9.1.2[55] give a more detailed view how to connect the pattern in a sequence and they contain the information when and how to proceed from one pattern to another pattern. We claim that this way we formed the grammar for our pattern language and published the claim in (Beckers, Faßbender, and Heisel, 2014 [50]).

As a result, we claim to have defined the vocabulary of a pattern-language via our meta model, form and pattern catalog, and the syntax via our defined relations between the context patterns. We rely on the relation investigation tables and Figure 9.6[56] as grammar for the pattern language. Nevertheless, our view of integrating different patterns using relation investigation tables as solution is novel and we have to discuss further with the pattern community if this in combination with Figure 9.6[57] satisfies as a grammar for our pattern language.

9.4. Summary

In this work we identified relations between context patterns using the meta-model described in Section 7.3[58]. The relations were investigated in detail using relation investigation tables we have introduced in Section 9.1.2[59]. The result of this work is a pattern language for context patterns with a detailed description of the relations.

We contribute the following in this chapter:

[47] Page 159
[48] Page 168
[49] Page 169
[50] Page 171
[51] Page 169
[52] Page 171
[53] Page 171
[54] Page 169
[55] Page 159
[56] Page 169
[57] Page 169
[58] Page 120
[59] Page 159

- We defined relation types and analyzed all relations between our existing context patterns. These relations form the syntax of our pattern language.
- We showed how context patterns can be combined to sequences of context pattern which solve a problem at hand. In this way we formed a grammar for our context pattern.
- These two elements, the syntax and the grammar, form a pattern language when combined with the meta model and pattern catalog introduced in previous chapters.
- We compared our pattern language with different existing definitions for pattern languages.
- We provided guidance for future context pattern providers who want to integrate their new context patterns into the context pattern language.
- We also provided guidance for context pattern consumers who want to combine different context patterns to describe the context for a problem at hand.

Chapter 10

Application of and Reflections on Context Patterns

This chapter serves the purpose to exemplify the application of context patterns using the running example as introduced in Chapter 5[1] as well as to present and discuss the validation case for the context patterns. This way, the reader gets an impression of the suitability of the context patterns for context elicitation. In Section 10.1 we show the application of the SOA Layer Pattern and the SOA Stakeholder Pattern to our running example. This application example enables the reader to comprehend how the usage of context patterns actually works. Next (Section 10.2[2]), we introduce the validation cases and respective research question which we used to assess the sufficiency of the context pattern for context elicitation. In Section 10.3[3] we discuss the results of the validation cases, the evidence they provide that context patterns fulfill the requirements for a context elicitation solution as defined in Section 6.3[4], and finally judge the sufficiency of context patterns for context elicitation. Last (Section 10.4[5]), we conclude this chapter.

10.1. Application to the Running Example

In this section we start to instantiate the context patterns of the context pattern sequence we identified in Section 9.2[6] for our running example Media Market (see Chapter 5[7]). The identified sequence of patterns is (see Figure 9.8): SOA Stakeholder Pattern, SOA Layer Pattern, Cloud System Analysis Pattern, Law Identification Pattern, and Law Pattern. Note that we will not discuss the instantiation of the Cloud System Analysis Pattern as this Pattern is not in focus of this thesis. Also the instantiation of the Law Pattern and the Law Identification Pattern is not part of this section. These Patterns and their instantiation will be discussed in detail in Chapter 14[8], Chapter 15[9], Chapter 16[10], and Chapter 17[11]. Hence, in this section we will only discuss the instantiation of the SOA Layer Pattern and the SOA Stakeholder Pattern.

Note that we do not start with the SOA Stakeholder Pattern as suggested by the pattern sequence, but with the SOA Layer Pattern. The reason is that for the joint use of the patterns

[1] Page 75
[2] Page 188
[3] Page 197
[4] Page 87
[5] Page 201
[6] Page 171
[7] Page 75
[8] Page 239
[9] Page 267
[10] Page 281
[11] Page 307

... The business idea of our example is to introduce a **content aggregator** as a mediator between **customers** and **content providers**. The aggregator collects the offers of different content providers. These offers are aggregated by the aggregator and then made available to the customers. The content aggregator also handles the payment by integrating **banks** into the business process. ...

Table 10.1.: *Important Part of the Example Description Containing the Organizations*

the pattern method of the SOA Stakeholder Pattern (see Section B.1.5[12]) suggests to start with the technical view provided by the SOA Layer Pattern. Hence, we switch the order.

Switching the order in this case does not invalidate the sequence as described by Figure 9.8 or the method for finding such a sequence. Following the original sequence is also possible and leads to the same result, but taking advantage of the suggested order and method lowers the effort for instantiating bot SOA patterns.

Note that we will mainly rely on analyzing the description of the running example for the instantiation presented in the following which is usually not the case in real setting. In a real setting, interviews and discussion are the main tools for getting the relevant information.

10.1.1. Instantiation of the SOA Layer Pattern

For instantiating the SOA Layer Pattern, we follow the method as described in Section 8.3[13]. The method consists of the steps *describe organizations*, *describe choreography and coarse grained processes*, *describe operational systems*, *describe components*, *describe business services*, and *describe infrastructure services*. The result of these steps is shown in Figure 10.1. We describe it in the following.

Describe organizations In the first step, we describe the important organizations. Table 10.1 shows the relevant text for identifying the important organizations as it describes the business idea and all organizations relevant for realizing this business idea. Hence, we collect the organizations *Customer*, *Content Aggregator*, *Content Provider* and *Bank* from the description in Chapter 5[14]. While the *Content Aggregator* only represents the specific aggregator for whom the SOA has to be developed, the other organizations represent groups. The *Content Aggregator* is the mediator between *Customer* and *Content Provider*. Hence, the business relations reflect this mediation. To accomplish payment, all organizations have to have a business relation to *Banks*.

Describe choreography and coarse grained processes Next, we have to structure and describe the choreography and coarse grained processes. Figure 5.1[15] already shows a high level

[12]Page 436
[13]Page 144
[14]Page 75
[15]Page 76

Content Look Up
... The aggregator collects the offers of different content providers. These offers are aggregated by the aggregator ...
... search for ... the desired content....
Content Delivery
... and then made available to the customers....
... and retrieve (A3) of the desired content....
Content Payment
... The content aggregator also handles the payment by integrating banks into the business process....
... the large number of different banks and online payment systems, which have to be integrated to fulfill the payment needs....

Table 10.2.: *Statements Giving Rise to the Different Processes*

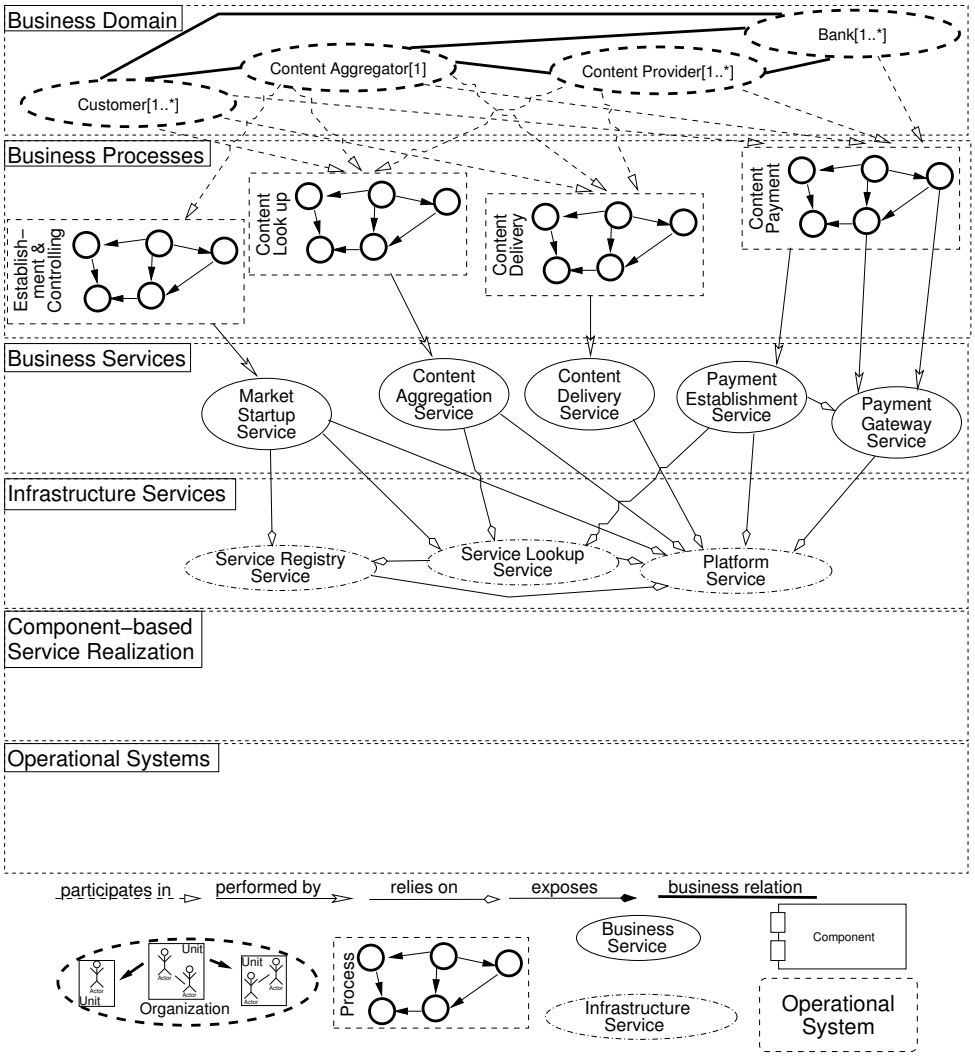


Figure 10.1.: *SOA Layer Pattern Instance for the Media Market*

choreography. But it only describes the start up and establishment of the Media Market (*content aggregator establishment*) and the starting point for the actual business process (*e-commerce establishment*). Hence, we keep the *establishment & controlling* as first major process, which contains the *e-commerce establishment*, but also ongoing activities for maintenance and accounting while the media market is operation. Additionally, we refine the e-commerce establishment for the business process part and find three additional major processes (Table 10.2 shows the statements which give rise to the different processes): The *Content Look up* process, in which *Customer*, *Content Aggregator*, and *Content Provider* take part, the *Content Delivery* process, in which also these three organizations take part, and the *Content Payment* process, in which all organizations take part. We refine the choreography and these major processes further and end up with an integrated process description as shown in Fig. 10.2. In this process, we left out the establishment steps as mentioned in Figure 5.1[16]. Hence, we focus on the actual business process.

The business process shown in Figure 10.2 should enable the customer to consume content offered by different content providers. The content providers want to be paid. The process is simplified at some points. For example, the payment part of the process is much more

[16]Page 76

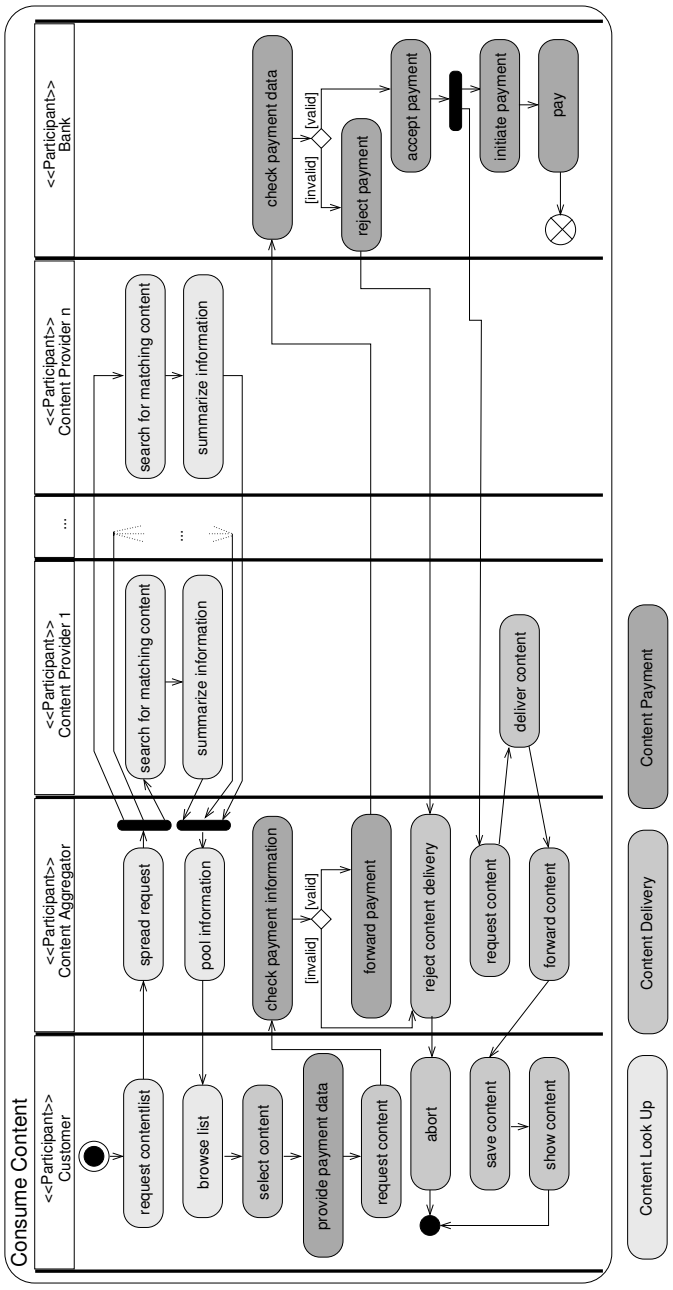


Figure 10.2.: *Coarse grained Business Process for the Media Market (UML Activity Diagram with SOAML-Profile Stereotypes)*

complex than shown in Figure 10.2. Note that the depicted process is modeled from a pure business perspective leaving out technical details such as, for example, the payment gateways as intermediary.

The process starts with a customer who *requests a content list*. This request contains some search criteria related to the desired content. The content aggregator *spreads this request* to all currently registered content providers. Each provider *searches for matching content* and *responds with a summary* of available content matching the search criteria. The summary also contains payment rules for the content they offer. The payment rules contain the pricing, accepted payment options and so forth. The summaries of the different providers are *pooled in one list and forwarded* to the customer. Next, the customer *browses the list* and *selects the*

desired content. In case the customer has to pay for the desired content he/she has to *provide payment data* for buying the content. If the customer is willing and able to pay, he/she *requests the content* from the aggregator. The aggregator *checks* if the *payment information* is valid. This means that a validity check is executed that decides if the order matches the payment data and if the payment data is correct regarding the payment rules of the corresponding content provider. The difference between payment data and payment information is that the payment information contains both, the payment data and the order data. In case it is *not valid*, the *content delivery is rejected*. Otherwise the *payment data is forwarded* to the according bank. The bank *checks* if the *payment data* is valid. In case the data is *invalid* the *request is rejected* and aborted. If the data is *valid*, the *payment is accepted*, *initiated* and finally *paid* by the bank. The bank also acknowledges the request, and the content aggregator *requests the content* from the corresponding content provider. Note that we simplified the process at this point as the consumer is able to request content from different providers, but we only show the case in which only one provider serves the content. This provider *delivers* the requested *content* to the aggregator, who *forwards* it to the customer. The customer *saves* and *consumes* the content.

Description of the operational systems and the components We skip the description of the operational systems and the components, because our scenario is a development of an SOA from scratch. Hence, no already existing technologies or systems are mentioned in the scenario description.

Describe Business Services The business processes of our example are already shaped in a way that each process has to be covered by an own business service. This way we get the business services *Market Startup Service*, *Content Aggregation Service*, *Content Delivery Service*, and *Payment Establishment Service*. In the case description we find one additional business service which encapsulates parts of the content payment process:

> ... *payment gateways*, which aggregate all banks and make payment functionalities available at a single point. These payment gateways offer their functionality also by services. (See Chapter 5[17])

Hence, we add the business service *Payment Gateway Service*. As the payment gateway is responsible for some activities of the original business process "Content Payment" as shown in Figure 10.2, we also update the business process. The resulting process is depicted in Figure 10.3.

Describe Infrastructure Services From the description of the e-commerce and content aggregator establishment (see Figure 5.1[18]), we can derive two infrastructure services. The statements

> ... content aggregator queries a service broker to find content providers ... and payment gateways ...
>
> The customers are now able to find the aggregators ...

give rise to a *Service Lookup Service*. The statements

> Content providers ... and payment gateways ... registered themselves at the service brokers ...
>
> ... content aggregators register their service at a service broker ...

give rise to a *Service Registry Service*. Additionally, the statement

> Cloud providers offer a scalable platforms for running services ...

[17] Page 75
[18] Page 76

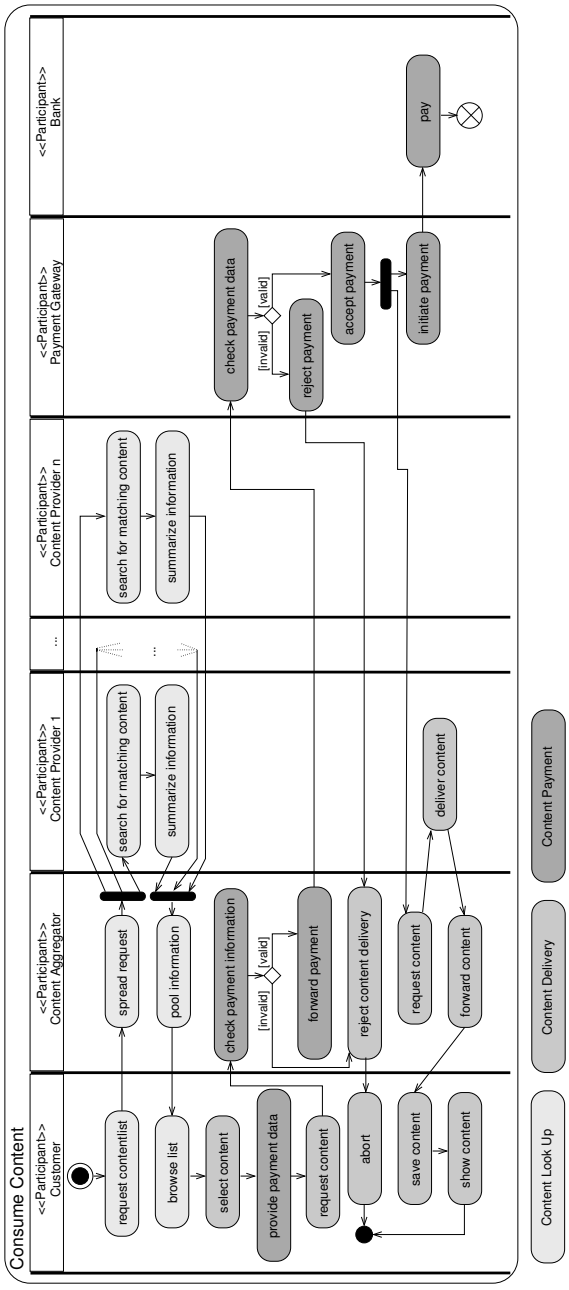


Figure 10.3.: *Coarse grained Business Process for the Media Market with 3rd Party Business Services (UML Activity Diagram with SOAML-Profile Stereotypes)*

indicates the existence of *Platform Services*. The business service *Market Startup Service* relies on the *Service Registry Service* as after a successful start up the Media Market has to be registered at a service broker to enable customers to find the Media Market. For finding the initial set of payment gateways and content providers the *Market Startup Service* relies on the *Service Lookup Service*. The business services *Content Aggregation Service* and *Payment Establishment Service* also rely on the *Service Lookup Service* as the list of content providers and payment gateways, respectively, will have to be updated from time to time. All infrastructure services are run on a certain *Platform Service*, which might differ for each business service. All in all, we obtain the instance of the SOA Layer Pattern as shown in Fig. 10.1.

10.1.2. Instantiation of the SOA Stakeholder Pattern

For instantiating the SOA Layer Pattern, we follow the method as described in Appendix B.1.5[19]. The result of the stakeholder elicitation phase of our method (see Figure B.15[20] for the method) is depicted in Figure 10.4. For the *Direct Environment*, we identified four process actors[21] and three providers. The process actors are derived from the process depicted in Figure 10.2. *Customer*, *Content Provider* and *Bank* are stakeholders representing a group. The members of these groups can change dynamically in our scenario, and the members are not that homogeneous that we can refine them further, for example, into roles. For the *Content Aggregator* we are able to do so. Here, we get two refined roles. The *Administrator* is responsible for setting up the system and controlling the technical aspects of the Media Market, while the *Accounter* is responsible for controlling the financial aspects such as billing and so forth. *Payment Gateway*,

[19] Page 436
[20] Page 445
[21] Note that even though the content provider has provider in its name, the content provider is a process actor from the perspective of the pattern.

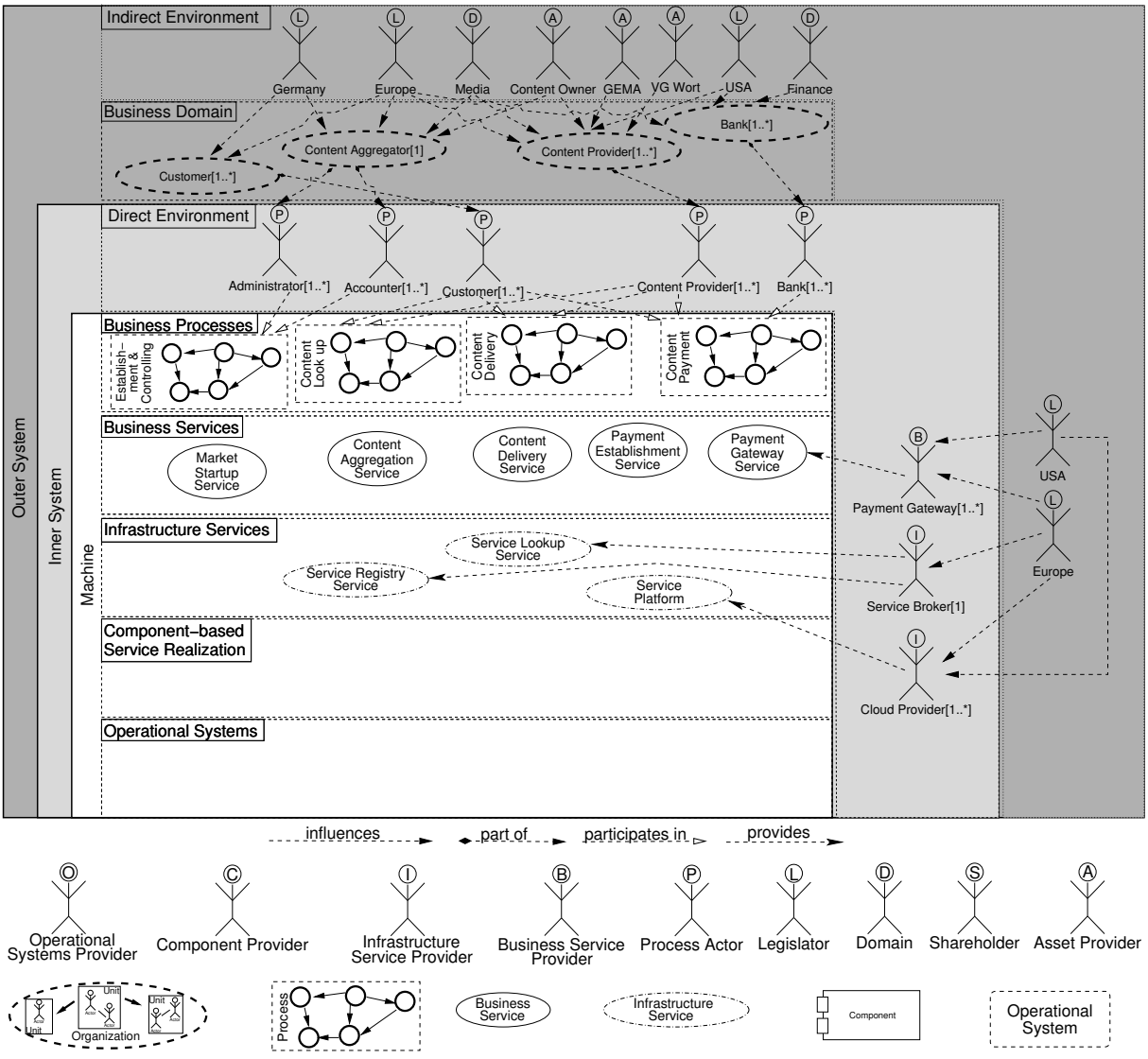


Figure 10.4.: *SOA Stakeholder Pattern Instances for the Media Market*

Payment Gateway (Business Service Provider)
. . . *payment gateways*, which aggregate all banks and make payment functionalities available at a single point. These payment gateways offer their functionality also by services. . . .
Service Broker (Infrastructure Service Provider)
. . . content aggregator queries a service broker. . . . . . Content providers . . . and payment gateways . . . registered themselves at the service brokers . . . . . . content aggregators register their service at a service broker . . .
Cloud Provider (Infrastructure Service Provider)
. . . Cloud providers offer a scalable platforms for running services.. . .

Table 10.3.: *Statements Giving Rise to the Different Providers*

Service Broker, and *Cloud Provider* were already mentioned in the initial unstructured scenario description (see Table 10.3), but as there is no decision for a specific provider up to this point, they also represent groups.

For the *Indirect Environment* we identify *Germany*, *Europe* and the *USA* as legislators, *Media* and *Finance* as domains, and *GEMA*, *VG Wort* and *Content Owner* as asset providers.

In our setting, the content aggregator wants to serve the German market, so it is likely that also the customers are from Germany. Thus, the legislator *Germany* has to be introduced. And as Germany is a part of Europe, *Europe* has also to be introduced. Since the big media companies reside in the US, we also add *USA*. To add *USA* is also necessary for the *Payment Gateway* and *Cloud Provider*, because some important gateways and cloud providers reside in the US. For the *Service Broker*, we take the decision to aim for a European one.

In Germany, there exist two special kinds of asset providers, besides the *Content Owner*, which directly plead themselves against content provider and aggregator. *GEMA* and *VG Wort* are right distributors, who plead all media owners against media consumers in Germany.

The content aggregator and the content provider are part of the *Media* domain. *Bank* is part of the *Finance* domain.

10.2. Validation

The sufficiency of the context patterns regarding the tasks of context elicitation and documentation was assets using different strategies and cases. Table 10.4 gives an overview of the strategies and cases applied for the different patterns. The strategies are given in the columns, the context patterns in the rows, and different validation cases are given in the table cells (A, B, and so forth). Note that in the following also cases regarding the Cloud System Analysis Pattern are reported, while the pattern itself is not topic of this work. But still the experiences and observations regarding the Cloud System Analysis Pattern contribute to the overall assessment of the context pattern idea. A short description of the validation cases is provided in the following (For a more detailed description see Appendix A[22]):

A The topic of this experimental *feasibility study* was to get a first impression whether the Cloud System Analysis Pattern and the Law (Identification) Pattern were usable at all

[22]Page 377

Context Pattern	Feasibility Experiment	Experimental Simulation	Case Study	Action Research
Cloud System Analysis Pattern	A		B	C
Meta Pattern				D
Law (Identification) Pattern	A	E		C
Smart (Grid / Home) Pattern				D
SOA (Layer / Stakeholder) Pattern	F	G	H	

Table 10.4.: *Overview of Empirical Validation Cases for the Context Patterns*

and sufficient for describing a cloud setting and find laws which are of importance for this setting. As these patterns were completely untested, such a test run was necessary to find flaws and gain first insights in their usability and usefulness. The experiment should provide insights whether a cloud setting can be described using the Cloud System Analysis Pattern, the information described by the Cloud System Analysis Pattern can be used for further activities, and if the Law (Identification) Pattern work as intended. The case was a banking scenario derived from some real cases we came across when discussing this topic with several companies. In the derived case, a bank wants to migrate paying services to a cloud provider. Several concerns in this context have to be assessed such as costs, security, reliability, and compliance. To judge compliance possibly relevant laws have to be identified. The law used within this case was the federal data protection act. For this validation case, the patterns were instantiated in a purely manual way without any tool support.

B In this *case study*, we investigated whether the Cloud System Analysis Pattern is suitable for a small enterprise to document its cloud infrastructure. This way, we tried to learn if the Cloud System Analysis Pattern is useful and usable for such a company. The assumption behind this case study is that when such a small company can successfully use a context pattern, the effort spent for using the context patterns can be spent by almost every company. The case study served two purposes. First, to assess a context pattern in a real scenario without any influence of the authors of the context patterns. Second, to investigate if a company with very limited resources in money, time, and people is able to use the context patterns or not. The case was a small enterprise offering a cloud infrastructure and cloud based services in the logistics domain. Here, the Cloud System Analysis Pattern should be used for threat assessment and, finally, an ISO27000 certification. This case was part of a three year long project in which also tooling for the Cloud System Analysis Pattern and the ISO27000 documentation was developed. Beside our working group and the small enterprise, a second small enterprise specialized on certification, and another working group specialized on security assessments were involved in this project.

C In this *action research*, we replicated Case A in a real setting, because the results from Case A were promising but needed validation in a real case. This action research should provide insights whether a cloud setting can be described using the Cloud System Analysis Pattern, the information described by the Cloud System Analysis Pattern can be used for further activities, and whether the Law (Identification) Pattern work as intended. The practical purpose was to write a security and compliance concept. The case was a cloud platform developed by a research institute. The cloud platform offers several services and solutions for companies in the field of logistics. First cooperations with companies from the industry were already tarted, but for going productive, those companies were in need of a security and data protection concept for the used services, which did not include a detailed analysis of the covered requirements. For this case, the federal data protection act of Germany was of particular importance.

D In this *action research*, we had to understand the smart grid domain. For this domain, we had no context pattern. Hence, we used the Meta Pattern to derive a new one. The new pattern was used to support the first steps of the Security Development Life Cycle (SDLC) by Microsoft, which was used by the project partner. The action research served two purposes. First, we investigated whether the Meta Pattern is helpful for deriving a new Context Pattern and whether the meta model contained in the Meta Pattern is able to express all entities of the newly derived context pattern. Second, we assessed the sufficiency of the newly derived patterns for collecting the information as demanded

by the initial steps of the SDLC. One application domain of the NESSoS project, which we took part in, is the smart grid. Our task within this project was to deliver solutions to analyze the smart grid regarding security, privacy and compliance, for example with standards, in the early phase of the software development life cycle. In this particular case, the industrial partner was a world wide operating company focusing on the areas of electrification, automation and digitization. The topic of the case was to improve the early steps of the SDLC, as the industry partner was in need of more guidance for these steps.

E The topic of this *experimental simulation* was to assess the Law (Identification) Pattern and the accompanied problem frame integration (transformation cards) using a real case for which the requirements as well as the final judgment of a court is known, and publicly available. The experiment should provide information about the precision and recall when using Law (Identification) Pattern instances as well as the effort spent on creating and using the patterns. The case was an actually developed e-voting system. By its very nature, the field of electronic voting is an interdisciplinary field, where legal and computer scientists work together. During the development of the first voting system used in Germany, this fact was neglected or inadequately considered. Hence, the federal constitutional court of Germany judged in 2009 that using this system for voting in 2005 was unconstitutional. The requirements specification of this voting system is publicly available as Common Criteria Profile for voting systems. So is the final judgment of the federal constitutional court of Germany.

F The topic of this experimental *feasibility study* was to get a first impression whether the SOA (Layer / Stakeholder) Patterns were usable at all and sufficient for describing an SOA setting. As these patterns were completely untested, such a test run was necessary to find flaws and gain first insights in their usability and usefulness. The experiment should provide insights whether an SOA setting can be described using the SOA (Layer / Stakeholder) Pattern, and whether the usage of the patterns was convenient. The case was the same case as used as running example in this work (See Chapter 5[23]). The case was described before any of the context patterns were existing.

G The topic of this *experimental simulation* was to get a first impression whether the SOA (Layer / Stakeholder) Patterns were usable for someone who is not an expert in the field of SOA, and whether the collected information is really helpful for conducting requirements engineering for an SOA-based system. The case was the same case as used as running example in this work (See Chapter 5[24]). The task for this experiment was to actually develop the system described using a rigid development process. The development was done by a student during his master thesis.

H The topic of this *case study* was to get evidence whether the SOA (Layer / Stakeholder) Patterns are suitable to improve the software development for a real system or not. In this case an already existing system, which was built in an ad hoc manner, should be replaced with a new, systematically developed one. We were interested if the SOA (Layer / Stakeholder) Patterns help to structure the redevelopment and improve the outcome. For this case, we decided to not involve ourselves in the actual development, but be pure observers to see how the patterns work in a real life setting, applied by the actual target audience of the patterns. The case was provided by a student of ours who was also involved in a small size company providing tailored SOA based solutions for customer relationship and enterprise resource management. As it is often the case for start up companies, the

[23] Page 75
[24] Page 75

development of the product was done in an unstructured, ad hoc manner. They had chosen SOA for the benefits on the technology level, but were unaware of its implications. They also were missing an overview of stakeholders and their influence on the system. As the customer base was growing and also the usage of the old product, it became obvious to the small company that the established system was not maintainable on the long run as too many requirements were completely overlooked in the beginning. This led to the situation that they planned to replace the old system by a complete rebuild. For this case, we provided the student a description of a development method he could use, and the patterns. He was then asked to use them and modify them as he wished to adapt them to the companies' preferred agile development cycle.

A more detailed (but still brief) discussion of the validation cases can be found in Appendix A.1[25] and Appendix A.2[26].

Of course, these validation cases were conducted to answer particular research questions. In the following, we will discuss those research questions and corresponding answers, which are relevant regarding the context patterns.

A RQ1.A Is the Cloud System Analysis Pattern suitable to collect all important information in a structured form? In a first iteration within this feasibility experiment, we had only the graphical representation of the context pattern. But using the graphical representation alone, we were not able to document all necessary information. Hence, at this point the Cloud System Analysis Pattern was not sufficient to collect all important information in a structured way. In consequence, we extended the context patterns with information collection templates. In the next iteration, much more information could be documented using the patterns. Still, detail information such as technical details (for example, the specification of nodes within the cloud), or fine grained role models for an organization were left out. But we observed that these information was not of relevance for the further activities which should be enhanced using the context pattern. *Hence, the Cloud System Analysis Pattern helps to collect all important information.*

RQ2.A Is the Cloud System Analysis Pattern convenient to use and the effort spent for using it reasonable? All participants regarded the Cloud System Analysis Pattern as convenient to use. The graphical part was especially regarded as convenient to use for establishing a first understanding and to maintain an overview while discussing details. The information collection templates were regarded as handy to capture the results of the discussions about one particular element of the Cloud System Analysis Pattern. The effort spent for instantiating the Cloud System Analysis Pattern was about 40 person hours, which was regarded as reasonable for such a task. Even when keeping in mind that in a real setting more participants might be involved in instantiating the pattern, the effort was considered to be reasonable, as all participants expected that a one day workshop involving all necessary participants is reasonable. *Hence, the Cloud System Analysis Pattern is convenient to use and the effort to be spent is reasonable.*

RQ3.A Is the collected information suitable for a law identification? We observed that the participants were able to obtain some very general, early requirements, which still lead to a successful identification of the federal data protection act. But the information was not suitable to directly formulate fine grained requirements and, as a consequence, only the law as such was matched but not particular sections within

[25] Page 377
[26] Page 381

the law. Hence, it was difficult to derive the necessary requirements to make the cloud system compliant. As a result, we decided to integrate the context patterns with a requirements engineering method which allows one to decompose an overall problem to more fine grained sub-problems (see Chapter 15[27]). *Hence, for a first general identification of laws, the information is suitable, but for a detailed assessment the Cloud System Analysis Pattern cannot be used alone.*

RQ4.A Is the combined use of Law Pattern instances and Law Identification Pattern instances sufficient to identify a relevant law? The matching of Law Pattern and Law Identification Pattern instances was straightforward and successfully identified the federal data protection act. But we have to keep in mind that at this time we were only looking at the federal data protection act. *Hence, we obtained an indication that Law Pattern instances and Law Identification Pattern instances are sufficient for identifying relevant laws.*

RQ5.A Are the Law (Identification) Pattern convenient to use and the effort spent for using them reasonable? The effort for instantiating the Law Pattern for a law was regarded as considerable (about 40 person hours for the federal data protection act) but reasonable when keeping in mind that a law has to be treated only once and the instances are reusable afterward. The only drawback of the manual instantiation without tool support was the problem of keeping the instances for the different sections of one law coherent. The effort spent for the instantiation of the Law Identification Pattern was regarded as high. The effort spent accounted to about 50 person hours, which was considered as on the edge for being reasonable. The matching itself took about 12 person hours. The matching was considered to be really tedious. The instantiation of a single Law (Identification) Pattern was observed to be convenient for the participants. *Hence, the Law (Identification) Pattern is convenient to use for a single instantiation, but the accumulated amount of effort spent for the Law Identification Pattern instances is regarded as being not reasonable when doing the instantiation without tool support. The same applies for the matching itself.*

B RQ6 Can the Cloud System Analysis Pattern be applied without involving a context pattern expert? The project members did not face any serious problems while instantiating the Cloud System Analysis Pattern. But the feedback was also that the pure pattern description was sometimes ambiguous and that the different examples provided to the project members served well to overcome these misunderstandings. The final result produced within the project was completely valid also from a context pattern expert's point of view. *Hence, the Cloud Systems Analysis Pattern can be applied without involving a context pattern expert if enough examples are provided.*

RQ7 Is the Cloud System Analysis Pattern regarded as useful by a small enterprise? Both, the project members of the small consulting company as well as the members of the small cloud provider company agreed that the results documented using the Cloud System Analysis Pattern were really useful for creating the certification documentation as well as an internal means for other activities such as planning new products, explaining the cloud to new customers and so forth. A strong point mentioned in this context was that using the pattern results in a unified wording for which each term is clearly defined. *Hence, these two small enterprises regard the pattern as useful, which is an indicator for the overall usefulness.*

RQ8 Is the Cloud System Analysis Pattern regarded as usable by a small enterprise? The employees of both companies considered the Cloud System Analysis Pattern

[27] Page 267

as usable. They highlighted the guidance provided by the pattern, the use of the graphical pattern instance as discussion means, and that the Cloud System Analysis Pattern helps to focus on the important aspects on the right level of abstraction. The only general concern was regarding the ability to adapt a pattern to company specifics such as, for example, the wording. The cloud system provider wanted to be able to adjust especially the wording, while the consulting company was interested in generating subpatterns for a domain. Within the project, tooling was developed to reflect these needs. The ability to easily adapt a pattern was regarded as surplus in usability afterward. Another concern was regarding the generality of the elements captured using the context pattern. While the elements as captured using the pattern were regarded as a very good starting point, both companies desired a way to refine some elements. For this project, the method for the Cloud System Analysis Pattern as well as the tooling were extended to reflect these needs. But all project members also acknowledged that these extensions were specific to their certification specific needs and are not necessary in general. In consequence, these extensions were not considered as a general part of the pattern. *Hence, the Cloud System Analysis Pattern is usable for small enterprises, which is indicated by the feedback of the two companies. A surplus is the ability to form subdomain or company specific patterns.*

RQ9 Is the effort spent for using the Cloud System Analysis Pattern regarded as reasonable? The effort spent (about 2 person months) was considered as reasonable by all project members and especially by the cloud provider. The main reason for this perception was the fact that using the best practices for generating certification documentation for the parts covered by the Cloud System Analysis Pattern took about 6 person months. *Hence, the effort spent for using the Cloud System Analysis Pattern is regarded as reasonable.*

RQ10 Can the Cloud System Analysis Pattern compete with current best practices in the field? The two documentations, the one generated by using the best practice in the field of ISO27000 certification and the other using the Cloud System Analysis pattern, were compared for answering this question. The result was that both documentations overlapped by more than eighty percent. While the information which was only obtained by applying the best practices was regarded as detail information which was not of further relevance, the additional information collected using the Cloud System Analysis Pattern contained information which was of central importance and which was just missed using the best practices. The main reason stated by the project members was the missing guidance by the best practices which leads to the situation that the project members were lost in the details while missing the overview about the important parts. The only drawback of the Cloud System Analysis Pattern mentioned was that the best practices rely on tools and notations the project members were used to while for the pattern they had to learn something new. But they also stated that the initial investment payed off as the effort decreased and the results improved. *Hence, using the Cloud System Analysis Pattern can compete with the current best practices and seems also to outperform them.*

C As this validation case was kind of a replay of validation case A in a real world scenario, we only highlight additional or differing information in the following.

RQ1.C Is the Cloud System Analysis Pattern suitable to collect all important information in a structured form? We can also conclude from this case that the collected information was complete in the sense that we did not have to elicit additional context

information in the steps afterward. The information collected using the Cloud System Analysis Pattern covered all information which was already documented. There was no existing important information which could not be documented using our context pattern. Moreover, using the pattern additional information was discovered, which turned out to be really important afterward for the security and compliance concept, and which was just overlooked. Another important aspect we observed in this case was that the different project members had different views and different wordings. The pattern helped us to find a common wording and to reconcile the different views on the system. *Hence, the Cloud System Analysis Pattern helps to collect all important information.*

RQ2.C Is the Cloud System Analysis Pattern convenient to use and the effort spent for using it reasonable? Here, we observed that our assumption drawn from case A, that the effort is reasonable even for a larger system with a larger number of stakeholders, was affirmed. Only a few number of discussions and interviews, and an initial textual description of the project were necessary to instantiate the Cloud System Analysis Pattern. For conducting the interviews, the Cloud System Analysis Pattern was regarded as really useful to structure the interviews and to ask the right questions. *Hence, the Cloud System Analysis Pattern is convenient to use and the effort to be spent is reasonable.*

RQ3.C Is the collected information suitable for a law identification? *For a first general identification of laws, the information is suitable, but for a detailed assessment the Cloud System Analysis Pattern cannot be used alone.*

RQ4.C Is the combined use of Law Pattern instances and Law Identification Pattern instances sufficient to identify a relevant law? For answering this question again only the federal data protection act served as an example, as time and budged did not allow to use the pattern-based method for more laws. In consequence, we were only able to affirm the observations made for case A. *Hence, we affirmed the indication that Law Pattern instances and Law Identification Pattern instances are sufficient for identifying relevant laws.*

RQ5.C Are the Law (Identification) Pattern convenient to use and the effort spent for using them reasonable? *The Law (Identification) Pattern is convenient to be used for a single instantiation, but the accumulated amount of effort spent for the Law Identification Pattern instances is regarded as being not reasonable when doing the instantiation without tool support. The same applies for the matching itself.*

D RQ11 Does the Meta Pattern support a context pattern provider when deriving new context patterns? For deriving the Smart Grid Pattern we used the method as described by the Meta Pattern successfully. The process included all steps which we had to conduct to get a new pattern which is accepted by smart grid experts. Additionally, the Smart Grid Pattern could be successfully integrated into the SDLC. Hence, it serves its purpose to elicit the context information which is required to conduct a software development life cycle. Moreover, the process of describing the new Smart Grid Pattern was less compared to the time it took to derive the former context patterns such as the SOA (Layer / Stakeholder) Pattern. The clear process avoided unnecessary iterations caused by missing information, which should have been provided by a preceding activity, within a later activity. Additionally, the method description clarified which inputs and outputs are required to finish an activity which eased the planning for deriving the new pattern. *Hence, the Meta Pattern supports in deriving new context patterns. Note that in this case only experienced context pattern providers were*

involved. Nevertheless, the result can be used as a weak indication that this might be true in general, too. But to strengthen this claim further research is needed.

RQ12 Does the meta model contained in the Meta Pattern contain all elements to express a new context pattern? *Yes, we did not find any element which had no corresponding element in the meta model (see Section 7.3.2[28]).*

RQ13 Is a context pattern derived for a domain using the Meta Pattern accepted by domain experts as valid? The Smart Grid Pattern as such was accepted by the domain experts. But again, we noted that the wording within a company can differ from the wording used in the original pattern. In the discussions with the experts, often the experts claimed that an element was missing and in all cases it turned out that it was already there but under a different name. Often this situation was caused by the fact that the experts tended to look only at the graphical notation, not reading the descriptions of the elements in detail. This wording problem can be fixed in discussions between the context pattern expert and the domain experts. Nevertheless, it turned out to be useful to adapt to a company specific wording, so that the pattern users of this company could easily take up and understand the pattern. Note that this wording problem does not invalidate the pattern itself. It is more a question how the use of a pattern can be eased within a company. Another thing we observed was that the experts were often talking only about a sub-domain. For such a sub-domain not all elements of the context pattern at hand might be necessary. This also does not invalidate the pattern itself, but highlights that there is a need for sub-domain patterns. *Hence, the Smart Grid Pattern derived using the Meta Pattern is accepted as a valid, general description of this domain.*

RQ14 Is the newly derived pattern suitable to guide the initial steps of the SDLC? We used the Smart Home Pattern, which is a sub-domain pattern derived from the Smart Grid Pattern for our case, to elicit and describe the information as required by the SDLC in the early steps. This information was regarded as sufficient for th SDLC. For more information on the integration and the results see Beckers, Faßbender, Heisel, and Suppan (2014 [51]). *Hence, the Smart Home Pattern is suitable to guide the initial steps of the SDLC.*

RQ15 Is the effort spent for using the context patterns acceptable with regards to the effort spent using the former methods? The industrial partner stated that the usage of the Smart Home Pattern eased the conducting of the early steps of the SDLC and speed up the execution of these steps. *Hence, the integration of the Smart Home Pattern lowered the effort spent in comparison to the former method.*

E For a more detailed discussion of this validation case see Chapter 17[29].

RQ16 Are the transformation cards sufficient to be integrated into the overall identification process as described by Beckers, Faßbender, and Schmidt (2012 [44])? *Yes, the integration created sufficient results.*

RQ17 Does using the transformation cards and the Law (Identification) Patterns lead to an identification of all relevant laws? *Yes, with a high precision and recall.*

RQ18 Are the transformation cards and the Law (Identification) Patterns competitive with a pure discussion-based assessment involving legal experts? *Yes, in this case the pattern-based approach even outperformed the discussion-based one.*

[28] Page 125
[29] Page 307

F RQ19 Are the SOA (Layer / Stakeholder) Patterns suitable to collect all important information in a structured form? In this feasibility experiment we observed that we could structure all information given by the case using the graphical representation and the information collection templates. Some additional information, which was not in the initial case description, was also collected. *Hence, the SOA (Layer / Stakeholder) Patterns were suitable to collect all important information in a structured form, and also indicated missing information*

RQ20 Is the use of the SOA (Layer / Stakeholder) Pattern convenient? While we started with the graphical representation to get an overview of the case, the information collection templates were of use when discussing single elements. In this phase, the graphical representation was used to maintain the overview, and to assure that the relations between the elements were correctly considered. *Both, the context pattern experts and the SOA and context pattern novices, considered the SOA (Layer / Stakeholder) Pattern to be convenient to use.*

G RQ21 Are the SOA (Layer / Stakeholder) Patterns suitable for a non-domain expert to collect all important information in a structured form? When comparing the instances the student produced and the instances we produced, we observed that the instance of the student was quite similar to ours. In most cases, only the naming differed. For the services we got differences regarding the granularity. The student tended to aggregate business services which are from the same organization. We had a more process oriented modularization. Both ways were accepted to be valid in the end, because the collected information remained the same. But we added the recommendation to the method to use a process oriented modularization. The reason is that later on the student also split up the business services according to the process. Another observation was that reading a graphical representation instantiated by somebody else can be confusing in case the different types of entities are not distinguishable. At this time all stakeholders and process actors were represented with the same stick-figure. Hence, we changed the representation to make the types distinguishable. *To conclude, a non-domain expert is able to collect all important information in a structured way using the SOA (Layer / Stakeholder) Patterns.*

RQ22 Is the use of the SOA (Layer / Stakeholder) Pattern convenient for a non-domain expert? The student reported that the patterns were easy to use. The only problem the student was facing was related to the indirect stakeholders. Here, the student observed that this concept as such is clear, but to keep on track and to identify the important indirect stakeholders is still challenging. *But overall, the student reported that the SOA (Layer / Stakeholder) Pattern were convenient to use and provided a good guidance for the task.*

RQ23 Is the collected information using the patterns suitable for improving downstream development activities? The student integrated context information collected using the patterns into the requirements elicitation, the requirements specification, and the design phase. He considered this information as very useful, in some cases even as crucial. The implementation of the system as well as the documentation of the development process and the created documentation were convincing for the reviewers. For most development steps the student also provided an argumentation why and how the context information improved the development process. *Hence, for the student the SOA (Layer / Stakeholder) Pattern was suitable for improving the downstream development activities.*

H RQ24 Do the SOA (Layer / Stakeholder) Patterns help to find all important information and

to structure the information? The student reported that the usage of the patterns revealed much information, which was only implicitly known by the company, but never made explicit. At this point the involved persons were surprised about the amount of information which turned out to be crucial to consider in the end, but which was never documented or even discussed before. Here, the student highlighted the usage of the patterns to structure and guide interviews. The created documentation using the patterns was regarded as well structured and covering all important aspects by the involved persons. The student also reported that the patterns helped to establish and maintain a common wording. In the beginning there were some ambiguities regarding the words used, which were resolved in the process of using the patterns. This was also regarded as important information and information structuring. *Hence, the SOA (Layer / Stakeholder) Patterns help to find all important information and to structure the information.*

RQ25 Is the collected information using the patterns suitable for improving downstream development activities which are conducted in an agile way? The student also reported that having this information in a structured way was supporting the understanding of the system by all involved persons as well as the information served the development as it provided a good starting point for especially the requirements elicitation, analysis, and partitioning for the agile sprints. Additionally, the pattern instances served as means to maintain the overview about the complete system while conducting a sprint focusing on one part of the system. The collected information was even integrated into the design of the system. *Hence, the downstream development activities conducted in an agile way were improved by the application of the SOA (Layer / Stakeholder) Pattern.*

RQ26 Are pattern users able to apply and use the patterns without any detailed guidance beyond the pure description? Regarding this question, the student reported no problems while using the patterns. He as well as the other involved persons from the company had no problems while using the patterns. They also stated that the examples provided helped a lot. A review of the produced documentation by the pattern providers also showed that the pattern instances were valid with regards to the intended semantics of elements and so forth. *Hence, pattern users are able to apply and use the SOA (Layer / Stakeholder) Pattern without any detailed guidance beyond the pure description. Examples are an important part of the description with regards to the understandability of the patterns.*

RQ27 Is the resulting new system regarded as an improvement compared to the existing system? The overall development was regarded as success, as the new system outperformed the former one, added new features, and the effort spent was significantly lower than expected in the beginning. The involved persons also assume that the new design of the systems improves the maintainability a lot. The system itself is operational right now. *Yes, it is.*

10.3. Discussion

When judging the context patterns as means for context elicitation, we first have to have a look at our requirements for a context elicitation method as described in Section 6.3[30]:

Shared vocabulary The answers to **RQ1.C**, **RQ7**, **RQ13**, and **RQ24** strongly indicate that the usage of context patterns helps to define a shared vocabulary. This observation meets

[30] Page 87

our initial expectation that using patterns in general helps to find a common terminology. From the validation cases we can also conclude that the vocabulary defined by the context patterns themselves has a sufficient expressiveness as we never got results which indicated missing terms.

Abstract and detailed information The answers to **RQ1.A**, **RQ1.C**, **RQ19**, and **RQ24** highlight the ability of the context patterns to capture, combine, and document abstract and detailed information in a satisfactory way. Hence, when instantiating a context pattern both kinds of information are treated. Additionally, the observations also strongly indicate that the more abstract overview in means of the graphical model as well as the information collection templates for capturing detailed information about an element are regarded as useful.

The answers to **RQ2.A**, **RQ6**, **RQ22**, and **RQ26** indicate that the pattern form helps to get a quick overview of a context pattern, but also contains the detailed information needed to use them. The only drawback of the actual pattern form we observed was that the pattern form itself does not contain an application example. But the pattern users often stressed the importance of such examples for the understanding of the context pattern at hand. Here, we are currently discussing whether the examples should be kept separated as done in this thesis or if they should be an integral part of the pattern form.

Reconciling different view points Here, we only had two validation cases (C, D) for which different view points were reported. For this cases, the answers to **RQ1.C**, and **RQ13**, respectively, indicate that context patterns are suitable to find, discuss, and reconcile different view points. For the other cases, the existence of different view points and the need of reconciling them were not mentioned explicitly. But in turn, we also did not get any results that differing view points were a problem when using the context patterns.

Iterations All the methods contained in the different context patterns actually support iterations. But we did neither get any explicit results indicating that iterations are supported well, nor results indicating problems when iterating. From our experience, we can tell that changing the graphical representation or even maintaining different versions for a while is not a big problem. Keeping the information collection template instances up to date, and maintaining their coherence is kind of a problem. But in most validation cases, the iterations happened on basis of the graphical representation, while the information collection templates were instantiated once when the view on the system to be was already stable. Tool support improves the maintenance of the template instances even more, but still there is potential for further improvements. One idea might be to add rules which, when executed, ensure the coherence of the different template instances.

Models and Documents All of our patterns combine graphical models as well as textual templates. The answers to **RQ1.A**, **RQ1.C**, **RQ19**, and **RQ24** highlight the fact that the combination of graphical model and information collection templates enable the context pattern users to capture, combine, and document abstract and detailed information in a satisfactory way. The observations also strongly indicate that the graphical model is useful to get an overview while the information collection templates are regarded as useful to comprehend details.

Sufficient level of formalization Only the answer to **RQ10** implies that the level of formalization used within the context pattern caused some extra learning effort, which might be a drawback. But in this context it was also stated that the effort payed off. For all other validation cases, we got no explicit results in favor nor against the level of formalization of the patterns. Hence, we can assume that the level of formalization is acceptable for

the pattern users and that there is no adoption barrier caused by the semi-formal notation for the graphical representation. Additionally, the graphical model is already formal enough to allow an analysis of the models (see Beckers (2014 [37]) for more information), transformation (see Chapter 15[31]), or matching of models (see Chapter 16[32]).

Lightweight notation The notation we use within the context pattern is a notation crafted for the special purpose and as lightweight as possible. For each pattern the notation

- only contains the necessary domain concepts, because a carefully derived and described pattern only contains those entities important for the pattern. The results from all cases (for example, the answers to **RQ1.A**, **RQ7**, **RQ1.C**, **RQ11**, **RQ13**, **RQ19 RQ21**, and **RQ24**) imply that no entity was missing and in most cases all existing entities were used. Hence, all of the entities are necessary.
- is consistent, because for no case the pattern users reported any inconsistencies. Regarding generality, the answer to **RQ22** implies that for non-domain experts the level of generality of the indirect stakeholders might be too high to easily use them. But we also observed that this drawback diminishes when involving representatives, for example customers, of the domain. Additionally, the different instantiations of the indirect stakeholders within the cases showed that making the indirect stakeholders more precise would lead to missing indirect stakeholders not anticipated, and would increase the number of entities to an extent where it becomes confusing for the pattern user. Hence, we can conclude that the generality at this point is necessary, even though instantiating the indirect stakeholders is regarded as challenging by non-domain experts.
- makes all elements distinguishable. In the beginning we neglected this requirement. In consequence, we observed (**RQ21**) that a representation including elements with different semantics but same representation indeed negatively impacts the usability of the patterns. Hence, we adjusted the notation of each pattern in a way that the elements are distinguishable.
- adopts a UMLish representation to improve the learnability. For example, activities look like UML Activity Diagram activities, stickfigures represent the same concept as in UML Use Cases, and so forth.

From the results regarding usability (answers to **RQ2.A**, **RQ8**, **RQ2.C**, **RQ20**, and **RQ22**), we can see that the notation was never criticized or mentioned in a way that it hinders usability. Hence, we can conclude that the context patterns in their actual form use a notation which is sufficiently lightweight regarding its purpose of representing the main entities of a domain.

Structured interviews The answers to **R2.C**, and **RQ24** report that the context patterns were successfully used within interviews and helped to structure the interviews.

Avoid getting lost For all validation cases we observed that the context patterns really help to keep on track. For no case the involved persons reported that they were lost or even misguided within the elicitation and documentation of the context information. Moreover, the answers to **RQ10**, **RQ14**, and **RQ15** imply that using a context pattern improved the context elicitation while speeding up the context elicitation at the same time. This is a strong indicator that context patterns help to focus on the important aspects of a system.

[31] Page 267
[32] Page 281

From this discussion, we conclude that the context patterns fulfill all the requirements. *Hence, context patterns can be regarded as good solution for context elicitation considering these requirements for a context elicitation solution.*

But eventhough all requirements are fulfilled, we should still take a closer look at the usefulness, usability, the effort, and the results produced and their impact on a software development to judge the sufficiency of context patterns to improve the context elicitation within software engineering. The reason is that the requirements might be only necessary but not sufficient for a good context elicitation method.

Regarding the usefulness, the answers to **RQ1.A**, **RQ3.A**, **RQ4.A**, **RQ7**, **RQ1.C**, **RQ3.C**, **RQ4.C**, **RQ11**, **RQ17**, **RQ19**, **RQ21**, and **RQ24** give a clear picture that the context patterns used were useful to elicit and document the necessary context information. *Hence, we conclude that the context patterns are sufficient to elicit and document context information.*

For usability we get almost the same picture. The answers to **RQ2.A**, **RQ5.A**, **RQ8**, **RQ2.C**, **RQ5.C**, **RQ20**, **RQ22**, and **RQ26** can be summarized to the conclusion *that the context patterns are usable for domain experts and non-experts, and that they are usable within SMEs as well as bigger companies.*

Considering the effort, we have to take a look at the answers to the questions **RQ2.A**, **RQ5.A**, **RQ9**, **RQ2.C**, **5.C**, and **RQ15**. Here, we get a differentiated picture. For the Cloud System Analysis Pattern and the Smart (Grid / Home) Patterns we get the result that these patterns can be used spending a reasonable amount of effort. They even outperform solutions used beforehand. For the Law (Identification) Pattern it is reported that instantiating a single instance is convenient, but the effort spent for maintaining the coherence between instances and the matching is beyond reasonable boundaries. Here, (semi-)automated tools are needed to support the pattern user. As reaction, we developed such a tool support, and its application reduced the effort significantly for the criticized tasks (see Chapter 17[33]). Also for the other patterns we observed that adding a tool for instantiating a context pattern lowers the effort of using them even more. For the validation cases in which we applied the SOA (Layer / Stakeholder) Pattern, we had no benchmark as comparison. But we did not get any complaints regarding the effort needed to use them, nor do our own experiences imply any shortcomings regarding the effort spent. *Hence, we can conclude that the effort spent for using the context patterns is reasonable. Appropriate tool support can even more lower the effort spent.*

The answers to **RQ3.A**, **RQ4.A**, **RQ10**, **RQ3.C**, **RQ4.C**, **RQ17**, **RQ18**, **RQ23**, and **RQ27** provide insights in the sufficiency of the results. For all assessed patterns we have evidence that the results produced are sufficient for their purpose. Moreover, they are competitive compared to actual best practice, and in many cases they outperform this best practice. The impact on a software development can be discussed considering the answers to **RQ14**, **RQ23**, and **RQ25**. For these validation cases, the integration with software development life cycles were successful, the documented information improved especially the requirements and design phases, and the final development results were considered as a success. *Hence, we conclude that using context patterns produces sufficient results, the patterns can be integrated in and improve a software development process, and they have a positive impact on the results produced by such a process.*

To sum up, we can conclude that we have collected evidence that using context patterns is a sufficient solution for context elicitation. Note that we are aware of the fact that especially the experiments were conducted on such a small scale, that they do not provide statistically significant results which are suitable for generalization. This is a weak point which should be improved by conducting larger experiments. But from our point of view the combination of the different case studies, action research, and experiments give rise to such an evidence which allow the generalization as presented in this section.

[33] Page 307

10.4. Conclusion

While reading this chapter, the reader should have gotten an impression of the suitability of the context patterns for context elicitation. For this purpose,

- we showed the application of the SOA (Layer / Stakeholder) Pattern to the running example,
- introduced eight validation cases including two feasibility experiments, two experimental simulations, two case studies, and two action researches,
- enumerated the related research questions, and the results regarding the questions,
- and we discussed the results of the validation cases with respect to the requirements for a context elicitation solution, and the general sufficiency of the context patterns for context elicitation.

Chapter 11

Goal and Process Elicitation

In this chapter, we show how to use existing notations and methods for the steps goal elicitation (Section 11.1), and process elicitation (Section 11.2). Section 11.1 is based on Alebrahim, Choppy, Faßbender, and Heisel (2014 [6])[1], and Section 11.2 is based on Faßbender and Aysolmaz (2015 [132])[2]. Section 11.3[3] concludes this chapter.

11.1. Goal Elicitation

In this section, we show the results of eliciting the goals of the different stakeholders for our running example. For eliciting and modeling goals, one can choose from many notations and according methods such as i* (Yu, 1996 [396]; Yu, 1997 [397]), Tropos (Bresciani et al., 2004 [73]), ARMOR (Engelsman et al., 2011 [128]), and KAOS (van Lamsweerde, 2009 [378]). There are also goal notations focusing on software qualities in general such as the NFR framework (Mylopoulos et al., 1999 [277]), or focusing on one quality, for example, security such as SI* (Massacci et al., 2010 [252]), and Secure Tropos (Mouratidis and Giorgini, 2007 [273]). Additionally, there are papers on different aspects of goal modeling (Horkoff and Yu, 2011 [188])[4]: Interaction detection between different functionality implied by goals (van Lamsweerde et al., 1998 [379]), conflicts between goals describing qualities (Beckers, Faßbender, Heisel, and Paci, 2013 [48]), satisfaction analysis (Amyot et al., 2010 [16]), metrics for goals (Franch, 2006 [150]), and so forth. Hence, there is a huge body of knowledge on goal modeling one can benefit from, and no gap is existing with regards to this topic.

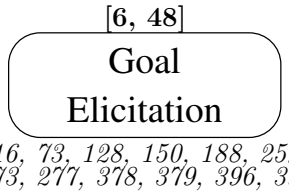


Figure 11.1.: *Goal Elicitation*

We followed the decision framework for selecting a goal notation as described by (Horkoff and Yu, 2011 [188]), ending up with i* as notation of choice. The application of the decision framework is shown in Appendix C[5].

Figure 11.2 shows the resulting i* goal tree for our example. Note that we will in the following

[1] The goal related content of this work is a contribution of the author.
[2] The UPROM related content is a contribution of Banu Aysolmaz and was revised by the author.
[3] Page 210
[4] Horkoff and Yu (2011 [188]) present an extensive survey of available papers in the goal oriented requirements engineering field. we just show examples.
[5] Page 469

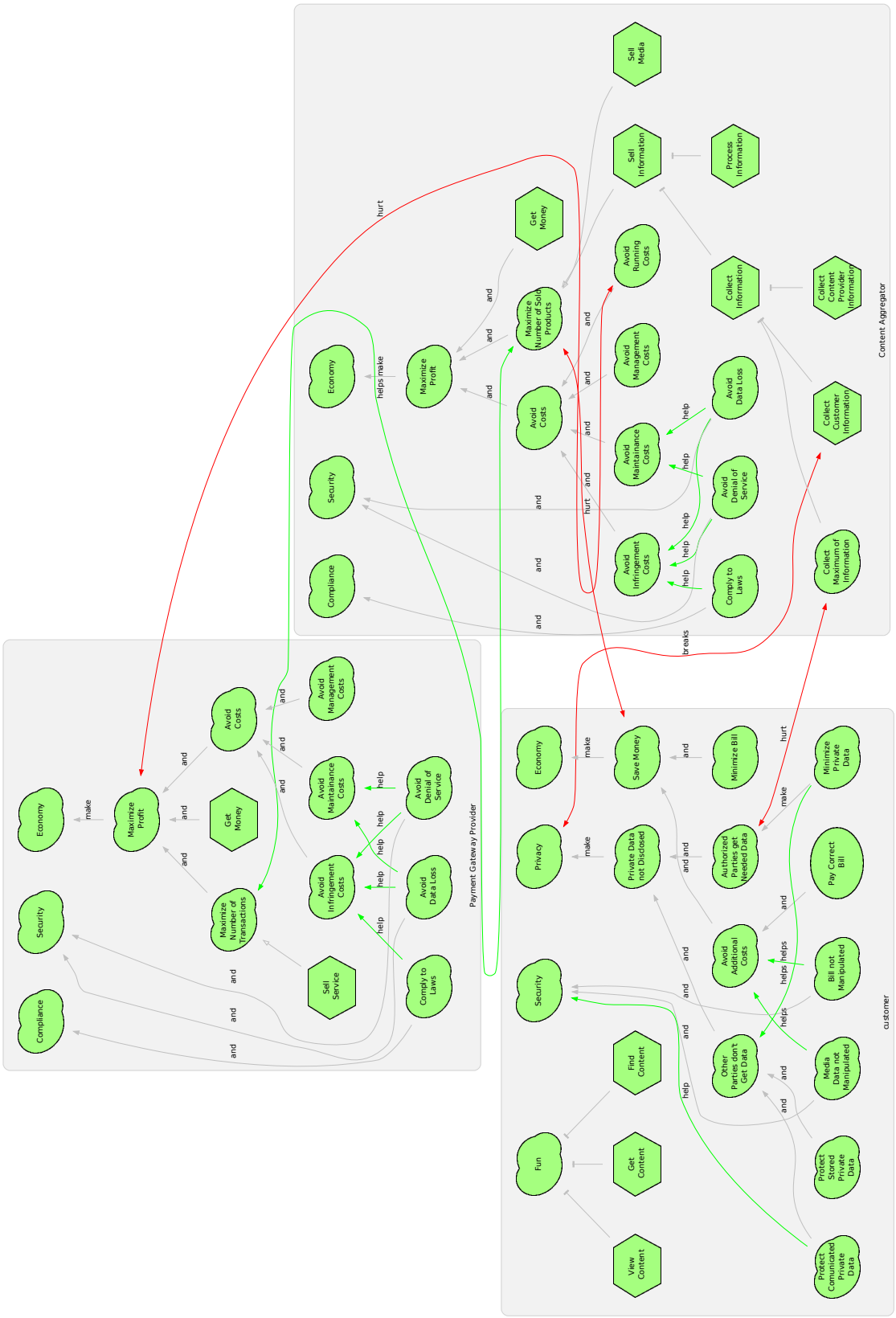


Figure 11.2.: *Goal Tree for Running Example (reduced)*

focus on the *Content Payment* process as already elicited in Section 10.1[6]. Hence, the goal tree is tailored to this process, and even further reduced to goals which are of central relevance for the following chapters[7]. We have done this for reasons of readability and comprehensibility. The full goal trees for the running example can be found in Appendix F.1.2[8].

The first actor shown is the *payment gateway provider*. According to the high level goals already elicited in Section 10.1[9] and documented in the corresponding context elicitation template (Table F.24[10]), the top level goals are *compliance*, *security*, and *economy*. To achieve the economic goal of the payment gateway provider, the *profit has to be maximized* which already fulfills (*makes*) the economic goal completely. To maximize the profit, the task *get money* has to be conducted, *the number of transactions* has to be *maximized*, and *costs* have to be *avoided*. To maximize the number of transactions the *service* has to be *sold* to others. To avoid costs in general, *management costs*, *maintenance costs*, *running costs* and *infringement costs* have to be *avoided*. For satisfying the security goal, the payment gateway provider wants to *avoid denial of service* attacks, and *avoid data loss*, which includes that the attacker obtains the data, due to attacks. Avoiding data loss and denial of service also *helps* to avoid maintenance and infringement costs. One particular goal relevant for the top level goal compliance is to be *compliant with laws*. Of course, this helps to avoid infringement costs.

The *customer* wants to achieve the top level goals *fun*, *economy*, *security*, and *privacy*. For having fun, the customer must be able *find*, *get*, and *view* content he / she likes. In order to achieve the *economy* goal, the customer wants to *save money*. Therefore, he / she has to *minimize the bill* for bought content, and *avoid additional costs* due to mistakes. To avoid additional costs, the customer has to *pay the correct bill*. The security goal is achieved when the *bill* and the *media data is not manipulated*. Both *help* to avoid additional costs as in case of manipulation money might be lost, or the paid content is unusable and has to be requested once more. Also the process of proving the manipulation for regression purposes might be costly. The privacy goal is made by the goal *private data not disclosed*. To assure this, *authorized parties should only get the data they really need*, and *other parties should not get any data at all*. To achieve the goal that no other parties are able to obtain data, the *stored data has to be protected* and the *communication of private data has to be protected*. This also helps to achieve the security goal. To ensure that authorized parties only get the necessary data, the customer wants to be able to *minimize the data actually sent*, which also helps to ensure that other parties do not get data.

The *content aggregator* wants to achieve *compliance*, *security*, and *economy*. To achieve economy, he / she has to *maximize the profit* of the media market. Therefore, *costs have to be avoided*, the *number of sold products has to be maximized*, and the *money* has to be collected. Costs can be *infringement*, *maintenance*, and *management costs*, which all have to be *avoided*. For maximizing the number of sold products, the media market has to *sell media*, but can also *sell information collected* while operating. One can *collect information* about the *content provider* and their offers, but also *collect information about the customers*, and in general the media market should *collect the maximum of information* possible. For satisfying the security goal, the content aggregator provider wants to *avoid denial of service* attacks, and *avoid data loss*, which includes that the attacker obtains the data, due to attacks. Avoiding data loss and denial of service also *helps* to avoid maintenance and infringement costs. One particular goal relevant for the top level goal compliance is to be *compliant with laws*. Of course, this helps to

[6] Page 181

[7] Due to the fact that the shown goal tree is just a snippet of the full goal tree, there might appear, for example, "and" relations where there is no second goal, and so forth. This is not a modeling error but a result of the reduction.

[8] Page 512

[9] Page 181

[10] Page 511

avoid infringement costs.

Indeed, there are also relations between the goals of the different stakeholders. The goal of avoiding running costs is of course in conflict with the goal of the payment gateway provider to maximize his / her profit, because a higher price for a transaction means more profit for the payment gateway provider, but also higher costs for the content aggregator. In turn, that the content aggregator wants to maximize his / her products sold helps the payment gateway provider to maximize the number of transactions, because each product sold results in a financial transaction. Moreover, there are some conflicts between the content aggregator and the customer. In case the content aggregator finds a way to sell the customer more products by, for example bundling media, and therefore maximizes the number of sold products, it hurts the goal of the customer to save money. The process of collecting customer information might hurt the goal of the customer to only provide necessary information, as the content aggregator might be motivated to request more information than he / she actually needs for the media selling purpose. But clearly, the collection of as much information as possible for selling the information breaks all the privacy goals of the customer.

11.2. Process Elicitation

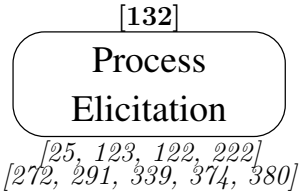


Figure 11.3.: *Process Elicitation*

Business process models, shortened process models, are blueprints of organizational processes (Kock et al., 2009 [222]). Optimization of business processes is found to be the key factor for cost efficiency and competitiveness within organizations (Bobs Guide, 2006 [60]). Many organizations are organized in a process oriented way. With the rise of process awareness and IT systems (Sinur and Hill, 2010 [356]), more and more organizations develop IT systems to automate their processes. These systems are referred to as process-aware information systems (PAIS). The percentage of European organizations that have an enterprise-wide business process management solution in use is more than 25%, and much more implement solutions on a smaller scale (Bobs Guide, 2006 [60]).

Business process modeling (BPM) is the key instrument to analyze and design a PAIS (Dumas et al., 2005 [122]; Monsalve et al., 2012 [272]). Process models of the business domain are used to reveal as-is process models, analyze them for weaknesses, and design to-be process models (Dumas et al., 2013 [123]). They help to communicate process knowledge among business users (Aalst, 2009 [1]; Indulska, Green, Recker, and Rosemann, 2009 [193]; Jamshidi and Pahl, 2012 [204]; Mayr et al., 2007 [262]). Process models are the main inputs to identify the user requirements of a PAIS (Vara et al., 2009 [380]). They are the drivers of PAIS, as the systems are developed to follow and support the modeled processes. However, process functions need to be further analyzed for their behavior, data usage and operations during PAIS execution to identify user requirements in the business domain (Aysolmaz and Demirörs, 2014 [27]; Nicolas and Toval, 2009 [283]; Monsalve et al., 2012 [272]; Vara et al., 2009 [380]).

Common business process modeling notations are (Sinur and Hill, 2010 [356]): Event-driven Process Chains (EPC) (Scheer, 1998 [339]), Business Process Modeling Notation (Object Management Group, 2011 [291]), and UML activity diagrams (UML Revision Task Force, 2012 [374]). Many commercial, open source and research tools as well as methods how to use these notations

are already available. Hence, in this field no gap exists and we make use of an existing notation and method. We have decided to use UPROM, an EPC variant, as it provides some benefits we discuss in the following.

UPROM is developed as a unified BPM methodology to conduct business process and user requirements analysis for PAIS in an integrated way (Aysolmaz and Demirörs, 2014 [25]). By following UPROM, business process and user requirements knowledge is captured in a set of models. These models are then used to automatically generate artifacts for PAIS development. These artifacts are functional user requirements, functional software size estimation and process documentation (Aysolmaz and Demirörs, 2014 [27]). To provide a seamless link between business process analysis and software development, methods to systematically utilize business process knowledge for PAIS development are necessary. UPROM already provides structured representation of knowledge in business domains in the form of models. UPROM models describe the functionality and conceptual data definitions for BPMS from a user perspective (Aysolmaz, 2014 [24]).

In UPROM, business processes are analyzed by developing extended event-driven process chains (EPC), value chain diagrams, function trees and organization chart diagrams. EPC, the core process modeling diagram for UPROM, is a popular process analysis notation introduced by the ARIS Framework (Scheer, 1998 [339]). ARIS is a popular and leading business process management tool in the industry (Norton et al., 2010 [288]; Sinur and Hill, 2010 [356]).

We followed the UPROM method as described by Aysolmaz (2014 [23]) to model the process as EPC diagrams, and the according functions appearing in the processes as FAD (for a description of the notation see Section 2.2.3[11]) . The processes are presented in Section 11.2.1 and the FAD diagrams in Section 11.2.2.

11.2.1. EPC Diagrams

The overall process, which is based on the coarse grained processes in the context elicitation step, and which relates these processes and adds some further activities is shown in Figure 11.4. Note that this process serves just as an overview and is only modeled in terms of events[12], functions,

[11] Page 33

[12] Note that trivial events can be left out of the process modeling. Hence, a function after a function is valid.

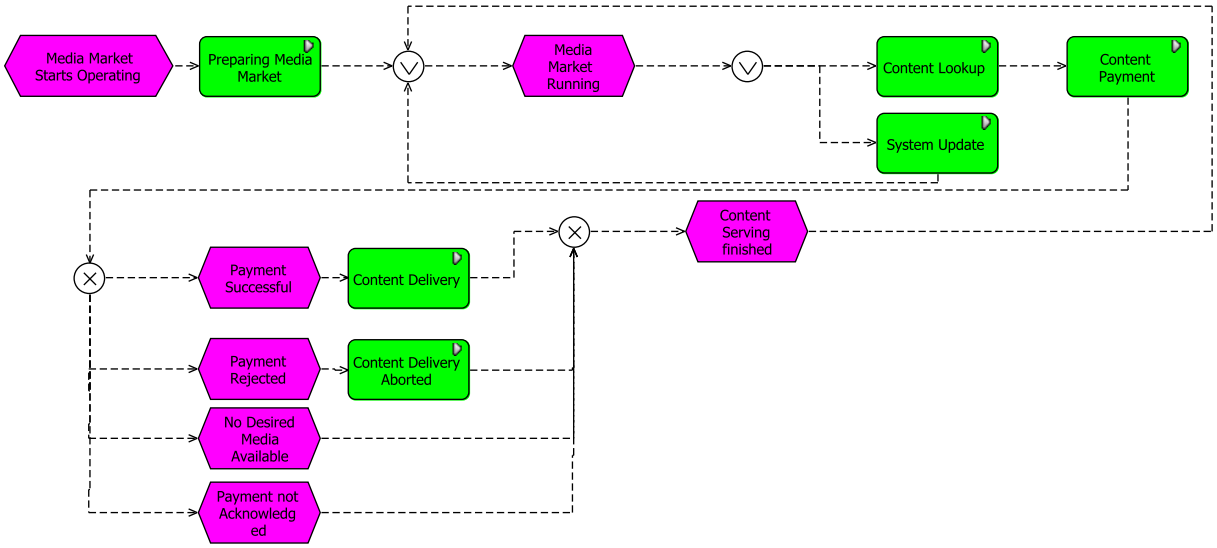


Figure 11.4.: *Overall Process Media Market*

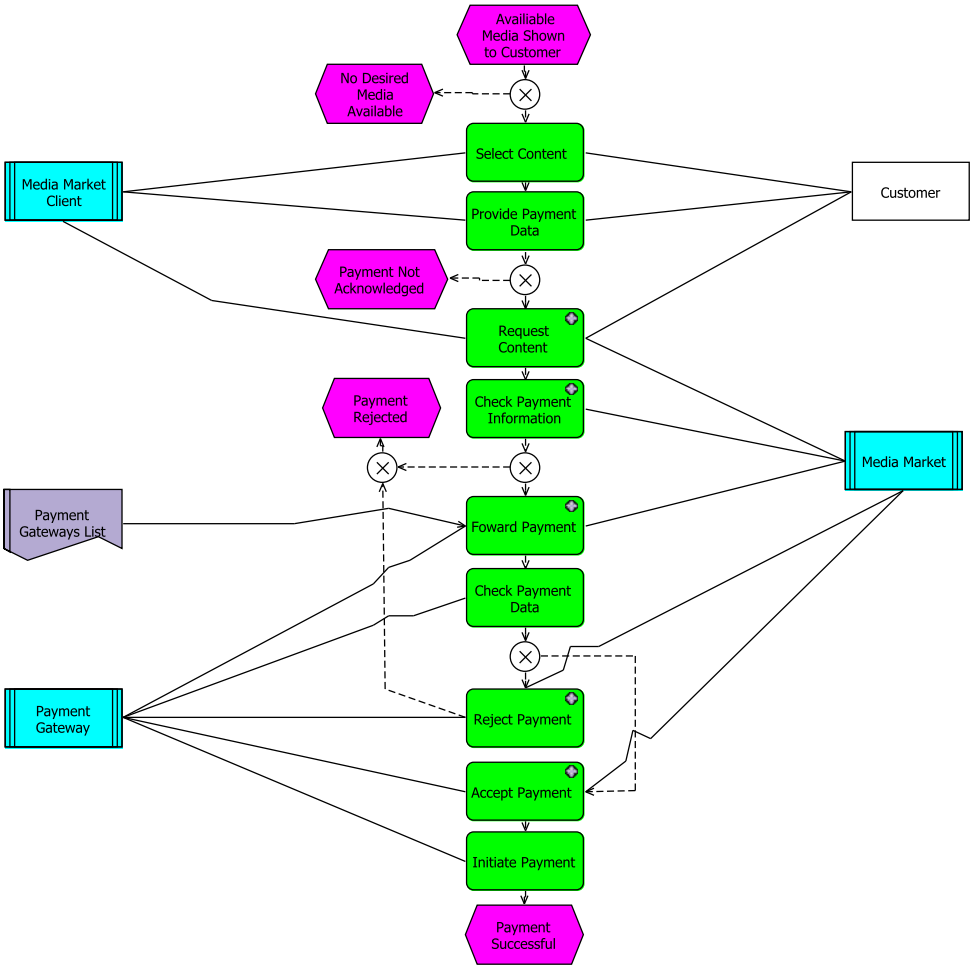


Figure 11.5.: *Content Payment*

gateways and control flows. The details are shown in the sub-processes. The process starts when the *media market starts operating*. The first activity [13] is then to *prepare the media market*, which means to add the initial configuration and data necessary for operating. Afterward, the *media market is running*. While running, one can initiate a *system update* to update the configuration, and so forth. After the update the system keeps running. Or a request for a *content lookup* occurs and the according sub-process is executed, and the *content payment* is initiated. In case the *payment is not acknowledged* by the user, or *no desired media was available*, the *content serving is finished*. If *the payment was rejected* for some reason, the *content delivery is aborted* by the media market, and the content serving finished. In case the *payment was successful*, the *content is delivered*. After the content serving is finished, the media market keeps running.

As already mentioned, we focus on the content payment process of our running example, for the sake of brevity, readability and comprehensibility. Hence in the following, we only show the according process modeled as EPC as well as the according FAD and ERD. But all EPC, FAD, and ERD diagrams are available in Appendix F.1.3[14].

The content payment process is shown in Figure 11.5. It starts when the *available media is shown to the customer*. In case *no desired media is available*, the process stops immediately. If there is desired media, the *customer* has to *select the content* he / she wants using the *media*

[13] Note that functions can represent complete sub-processes like in this case depicted by the triangular.
[14] Page 518

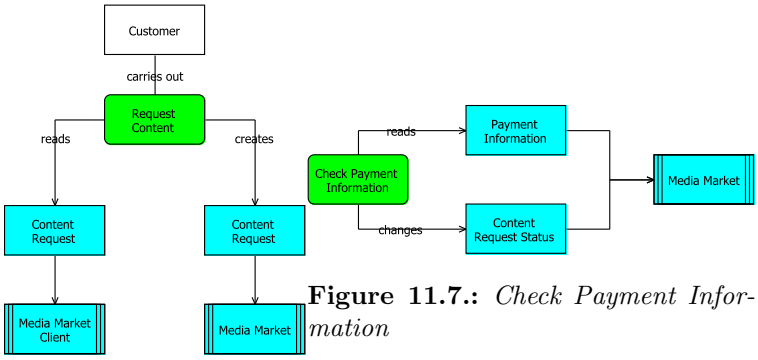


Figure 11.7.: *Check Payment Information*

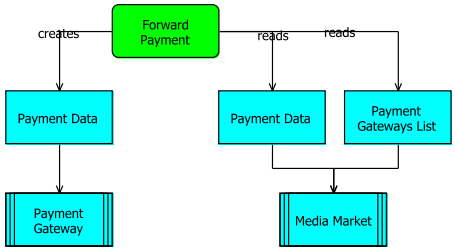


Figure 11.8.: *ForwardPayment*

Figure 11.6.: *Request Content*

market client. Then the customer has to *provide the payment data* to the client. In case he / she does not want to do this, the process is aborted because the *payment is not acknowledged*. If the customer provides payment data, he / she can *request the content* at the *media market* using the client. The media market makes then a first check of the payment information. In case the media market is able to determine its invalidity, the *payment is rejected*. In case the payment information seems to be valid for the media market, it *forwards the payment* to a *payment gateway* which is obtained form a *list of payment gateways*. The payment gateway *checks the payment data*. In case the *payment is rejected*, the payment gateway sends this information to the media market. In case the payment data is valid, the payment gateway *accepts the payment* and notifies the media market. Afterward, the payment gateway *initiates the payment* and the *payment is successful*.

11.2.2. FAD Diagrams

Following the UPROM method, all functions of the content payment process which involve the media market were refined using FADs. We discuss these FADs in the following, but first we have to discuss the relation between several information entities as depicted in Figure 11.11. The central piece of information is the *content request*. It consists of the *content request status* which indicates if the request is processed, rejected, and so forth, and the actual *payment information*. The payment information contains the *payment data* which is the information necessary for a payment such as credit institute, card number, and so forth, and the *order information* which contains information which media of which content provider is requested. We should keep the different information entities in mind, because they help to understand the FADs.

The *customer carries out* the function *request content* (Figure 11.6), in which the *content request* as sent by the *media market client* is *read* to *create* a content request to the *media*

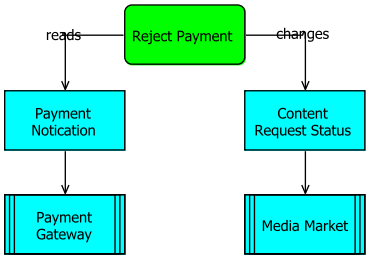


Figure 11.9.: *Reject Payment*

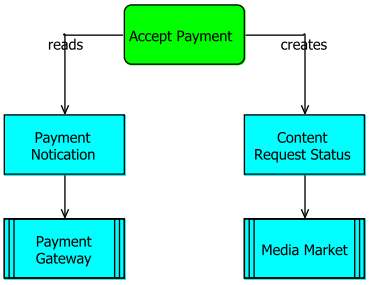


Figure 11.10.: *Accept Payment*

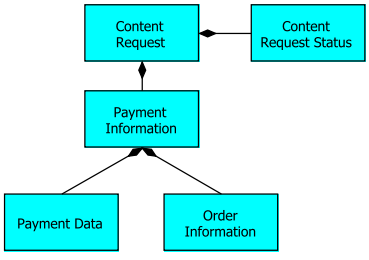


Figure 11.11.: *Content Request*

market. Note that according to the UPROM guideline a read operation on an entity which is related to an external application such as the media market client indicates that the information is sent by the external application.

The function *check payment information* (Figure 11.7) does not involve an organization element. Hence, the *media market* fully automatically *reads* the payment information to determine its validity, and to *change* the *content request status* accordingly.

For the function *forward payment* (Figure 11.8) the *media market* reads the *payment gateways list* in order to select a *payment gateway*, and then *reads* the payment information in order to sent (*creates*) the payment data to the payment gateway.

For executing the function *reject payment* (Figure 11.9), the *payment gateway* sends (*reads*) the *payment notification*, and the *media market changes* the *content request status*. In the same manner, the function *accept payment* (Figure 11.10) is executed.

11.3. Conclusion

In this chapter, we have analyzed the goals of the already elicited stakeholders as well as refined the process as elicited in the context elicitation step. Hence, now we know in detail what the stakeholders want to achieve when using the system. Additionally, we now have a complete picture of the processes the system-to-be has to support, which existing systems and stakeholders are involved in which step, and which information is of particular importance for executing a step.

In particular we have shown the results of applying

- i* to elicit and describe processes goals, and
- UPROM to elicit and describe processes

by applying them to our running example. For our running example, we have focused on the media market in general, and the content payment process in particular.

Part III.

Understanding the Problem

Chapter 12

Problem Context and Functional Requirements Elicitation

Starting with the step *problem context elicitation*, we switch from the stakeholder-centric view which focuses on the purpose of the system-to-be, to a system-centric view investigating the problem which shall be solved by the system-to-be. Within the phase *understand the problem* we have to turn the information collected in the preceding steps including stakeholders, existing systems, goals, processes, and so forth into system requirements which are suitable for deriving a specification for the system-to-be.

In Section 12.1 we discuss what important aspects for selecting a sufficient requirements engineering method and notation are, and we motivate for which reasons we have chosen the problem frame method and notation. Next, we show how to use the already modeled EPCs and FADs to ease the *problem context elicitation* step, and derive a context diagram from them (Section 12.2). Afterward, we show how to conduct the step *functional requirements elicitation* in the same manner, resulting in problem diagrams describing the functional requirements (Section 12.3[1]). Section 12.4[2] concludes the chapter. This chapter is based on Faßbender and Aysolmaz (2015 [132])[3].

12.1. A Method and Notation for Requirements Engineering

In order to accomplish the goal of specifying the system-to-be, we have to select an appropriate method and notation which allow us to describe, analyze and document the requirements. As already discussed in Section 1.4.3[4] we are using model-based notations and methods. Hence, also for this step we are looking for a method and notation allowing a model-based solution.

But which method and notation to choose? Curtis et al. (1988 [108]) already stated in the late 80ies that for developers writing the code is not a big problem, but understanding the problem which the code shall solve is a big problem for them. A field study in requirements engineering practices in the software industry of Australia by Sadraei et al. (2007 [332]) showed that requirements need to be represented in a way that they can be easily documented, coherently stored in one model at best, and that also the context and according domain knowledge is collected and described. Liu et al. (2010 [244]) made similar observations in a survey in the Chinese software industry. An experiment reported by (Burnay et al., 2012 [90]) showed that especially assumptions are an important part of the domain knowledge which has to be collected and documented in order to describe requirements sufficiently. Sutcliffe and Sawyer (2013 [363]) even state that the methods, notations, and tools for reasoning about requirements and according

[1] Page 217
[2] Page 219
[3] All content related to the actual transformation is a contribution of the author
[4] Page 19

assumptions is still one of the unknown unknowns, not sufficiently reflected in current research as well as practice. Espana, Gonzalez, and Pastor (2009 [130]) also stress the importance of describing the requirements in a way that they focus on the system itself and the problem it shall solve, describing its behavior within the environment. Furthermore, decomposition is seen as an important mean by Espana et al. (2009 [130]). This is seconded by other researchers (Ubayashi et al., 2008 [372]; Cheng and Atlee, 2007 [100]; Nuseibeh and Easterbrook, 2000 [290]; Zave, 1997 [398]; Jackson, 2001 [200]) as well as it is seconded by a study of Coughlan and Macredie (2002 [105]), which also revealed that in order to communicate requirements efficiently it is necessary to relate the requirements to their context, have a sufficient degree of decomposition of the original problem, and to represent the requirements in a uniform, easy to learn way. The importance of an easy to learn and clean notation is also stressed by Cheng and Atlee (2007 [100]). To summarize, a requirements engineering method and notation for describing system requirements should focus on the problem to be solved, allow an easy documentation, store all information in one model, also require to describe the context and according domain knowledge, allow tool-based analysis, be system-centric, use decomposition for partioning the overall problem, and be easy to learn.

Regarding these observations, the problem frames notation and method as proposed by Jackson (2001 [200]) appears to be a sufficient solution. Problem frames are system-centric, and embody a description of the environment of the system-to-be and the overall problem solved by the system-to-be. Using problem frames one decomposes the overall problem into small subproblems. Using the problem frames approach to define software requirements provides various additional benefits. First of all, it allows a thoroughly analysis and rigid definition of requirements (Hatebur and Heisel, 2010 [176]). Second, the context and according domain knowledge especially including assumptions can be modeled explicitly (Alebrahim et al., 2011 [5]). Third, we decided to use problem frames models because they have a semi-formal structure, which allows an (semi-automated) analysis of the requirements for, for example, different qualities, such as privacy (Beckers, Faßbender, Heisel, and Meis, 2012 [42]), security (Schmidt, 2010 [341]), and safety (Hatebur, 2012 [175]). Fourth, several topics such as aspect-orientation (Faßbender, Heisel, and Meis, 2014 [136]), and variability (Alebrahim, Faßbender, Filipczyk, Goedicke, Heisel, and Konersmann, 2014 [7]) can be treated if necessary. Fifth, using the UML4PF tool (Côté, Hatebur, Heisel, and Schmidt, 2011 [104]), the requirements are stored in one UML model, which can also store the results of subsequent activities. Sixth, from our experience the problem frame notation is easy to learn and use even by inexperienced students. Last but not least, problem frame models allow a seamless transition to the architecture and design phase (Alebrahim et al., 2011 [5]; Alebrahim et al., 2011 [4]). Hence, all the properties a requirements engineering method should cover for describing system requirements are met and there are even some further benefits.

12.2. Problem Context Elicitation

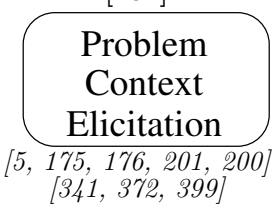


Figure 12.1.: *Problem Context Elicitation*

However in a normal case, problem frame models are modeled from scratch for each software development project and its requirements. This requires a considerable amount of effort and makes it difficult to establish the trace links between problem diagrams and business processes. But such a traceability is the key to ensure the completeness of

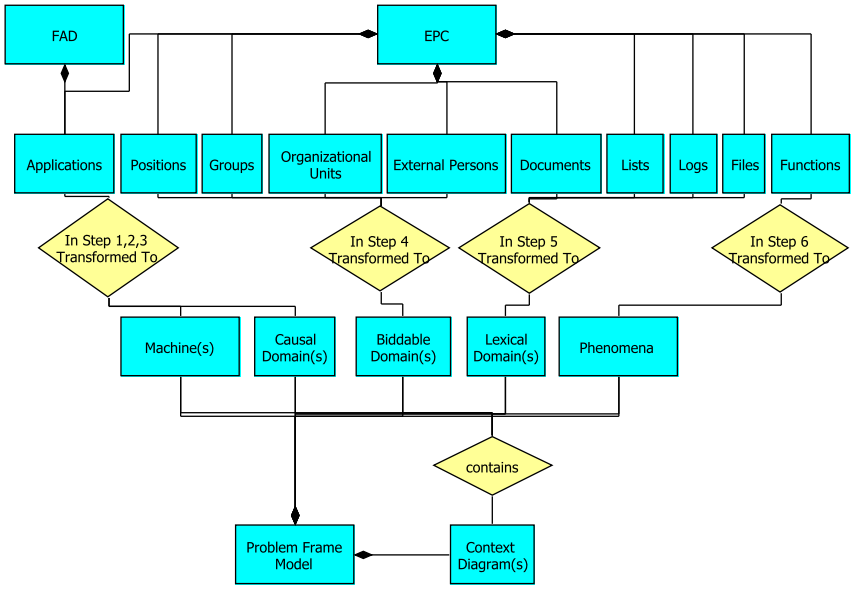


Figure 12.2.: *Relations between an EPC and Context Diagram(s)*

software requirements whenever the processes are already known, like in our case.

In consequence, we propose a solution to transform our UPROM business process models into a problem frames model. The central input for creating a context diagram is an *EPC* describing the business process and according sub-processes which shall be supported by the system-to-be (see Fig. 12.2). Elements of the types *application, position, group, organizational unit, external person, document, list, log, file* and *function* are part of the *EPC*. An exception are the applications which are not only obtained from the EPC but also from the related *FADs*. The reason for this exception is that in EPCs only the system under consideration is modeled explicitly, while other (external) applications are often only modeled explicitly in the related FADs. The expected output is an initial *Problem Frame Model*. *Machines, causal domains, biddable domains, lexical domains, phenomena*, and *context diagrams* are part of this model. A *context diagram* can contain all domains and phenomena which are part of a *problem frame model*.

The steps to transform the elements of the EPC into one or more context diagrams are explained in the following part.

1) Create Machines: In the first step we investigate the *applications* in the *EPC*. In case an application at hand has to be developed, we add a *machine* to our *problem frame model*. Decision rules for separating applications to be developed and external applications are described in (Aysolmaz and Demirörs, 2014 [25]).

2) Aggregate Machines: In many cases there are several *applications* to be developed, thus several machines, as part of an *EPC*. Now, we aggregate those to one *aggregated machine*. It might happen that there are several independent systems-to-be and therefore several *aggregated machines*. For each of them, we create a separate *context diagram*.

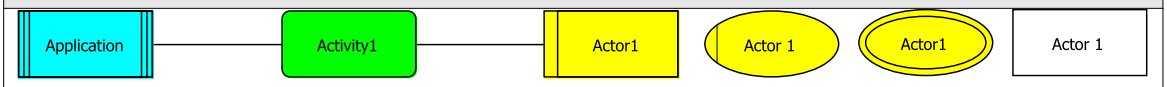
3) Create Causal Domains: For those *applications* which are not to be developed, we add *causal domains* to the *context diagram(s)*. We will refer to such applications as external or existing applications.

4) Create Biddable Domains: In this step, we consider elements which are of the types *position, group, organizational unit* or *external person*. Those elements have in common that their behavior is not fully predictable and they can be influenced only to some extent. Hence,

Case 1: Application, Function, Position | Group | Organizational Unit | External Person

Input

Part of the EPC

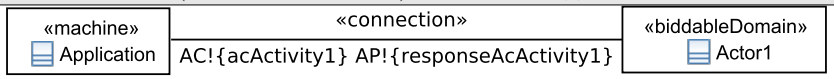


Questions (for semi-automated transformation)

(1)	Can the *Actor1* trigger the *Activity1*?	☐
(1.1)	Gets the *Actor1* a response by the *Application* when he/she triggers the *Activity1*?	☐
(2)	Can the *Application* trigger the *Activity1*?	☐
(2.1)	Gets the *Application* a response by the *Actor1* when it triggers the *Activity1*?	☐

Output

Option 2 (**1**☑ **1.1**☑ **2**☒ **2.1**☒): External Trigger with Response



(Default) Option 5 (**1**☑ / ☐ **1.1**☑ / ☐ **2**☑ / ☐ **2.1**☑ / ☐): Multi-Trigger with Response

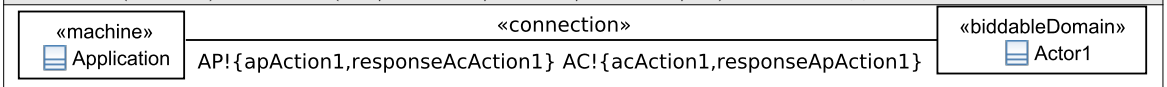


Table 12.1.: *(**Context Diagram**)Create Phenomena: Case 1 (Option 2 and 5 out of 5 Options)*

we turn them into *biddable domains* and add them to the *diagram(s)*.

5) Create Lexical Domains: In this step, we transform the *EPC* elements that represent information (*document, list, log* or *files*) to *lexical domains*. We then check if ERDs contain any information to aggregate them. If so, we add this joined element as lexical domain and create a mapping diagram for relation between that domain and its parts. We add all lexical domains which are not part of another lexical domain to the *context diagrams*.

6) Create phenomena: Only those functions that are connected to an application to be developed are used to create *phenomena*. *Phenomena* do not exist on their own but in relation to domains which control and observe them. Hence, we also have to take into account the other elements of the *EPC* connected to the functions. We distinguish four different cases for transformation depending on the type of elements connected to the functions. Table 12.1 shows parts of a transformation table for one of the cases[5]. The first part of the table defines the input required for the transformation. The first mandatory input is *part of the EPC* which defines elements and relations that have to be present in the EPC at hand to enable the transformation. Note that some elements can be exchanged: for example, the actor can be modeled using different element types. The *Questions* part defines the questions to determine the correct option for the transformation and is optional. The second main part defines the output. The output can differ as there are different options for the transformation. An option is described by the answers given (☑ stands for the answer yes, ☒ stands for the answer no, and ☐ stands for no answer or an indifferent answer). Underneath, the resulting output is given in the UML4PF notation. In case of a fully automated transformation, the default option as indicated in the transformation table is used, rather than expecting answers to the questions. Note that the complete transformation covers a combination of the four cases and different options. Hence, some phenomena might be created several times. In case a phenomenon already exists, it is not created a second time.

The resulting context diagram of the application of the EPC transformation to our running example is shown in Figure 12.3. The biddable domain **Administrator** controls phenomena (Interface A) to *configure the media market*, *the service broker* to be used, and to *retrieve payment gateways*. The machine **MediaMarket** controls the according responses. For *for-*

[5]All transformation tables can be found in Appendix D.1 (Page 471)

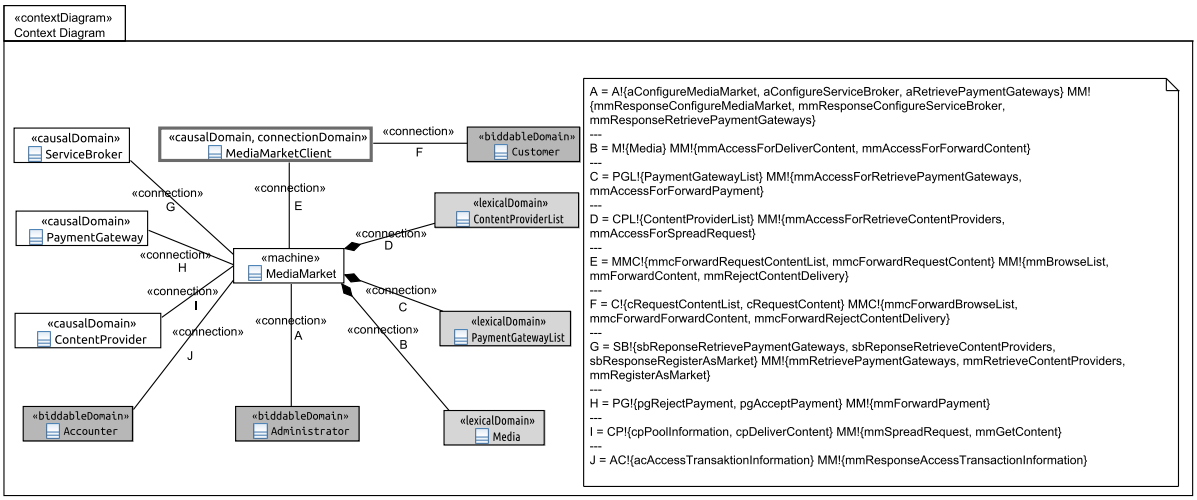


Figure 12.3.: *Context Diagram for the Media Market*

warding and delivering, the **MediaMarket** *accesses* the lexical domain *Media* (Interface B). For *retrieving the payment gateway list* and *forwarding payments*, the **MediaMarket** controls the according phenomena which are observed by the lexical domain **PaymentGatewayList** (Interface C). The lexical domain **ContentProviderList** is *accessed* for *retrieving the content providers* and *spreading requests* (Interface D). The biddable domain **Customer** is connected to the machine via the causal connection domain **MediaMarketClient**. The **Customer** controls phenomena for *requesting content lists*, and *requesting content* (Interface F). The **MediaMarket** can respond (Interface E) to the **Customer** using the phenomena for *browse list*, *forward content*, and *rejecting content delivery*. The **MediaMarketClient** forwards the phenomena of the machine and the **Customer** accordingly (Interfaces E and F). The **MediaMarket** issues the phenomena for *retrieving payment gateways, retrieving content providers*, and *registering as market* to the causal domain **ServiceBroker** (Interface G). The **ServiceBroker** responds accordingly. The machine also controls phenomena for *forwarding payments* to the causal domain **PaymentGateway** which in turn controls phenomena to *reject* and *accept payments* (Interface H). To *spread requests* and *get content* the *MediaMarket* controls the according phenomena which are observed by the causal domain **ContentProvider** which in turn controls the phenomena for sending *pooled information*, and *delivering content* (Interface I). The biddable domain **Accounter** can *access the transaction information* (Interface J). Hence, the context diagram in Figure 12.3 shows the machine in its direct environment which includes all important domains and phenomena as derived from the processes which shall be supported by the media market.

12.3. Functional Requirements Elicitation

The input for the process of *creating problem diagrams* are the *FADs* modeled in the process elicitation step. For each FAD we create a problem diagram including the according *requirement*. For each *application* which does not already exist and is part of the corresponding FAD, we add a new *machine* to the

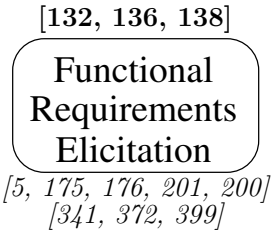


Figure 12.4.: *Functional Requirements Elicitation*

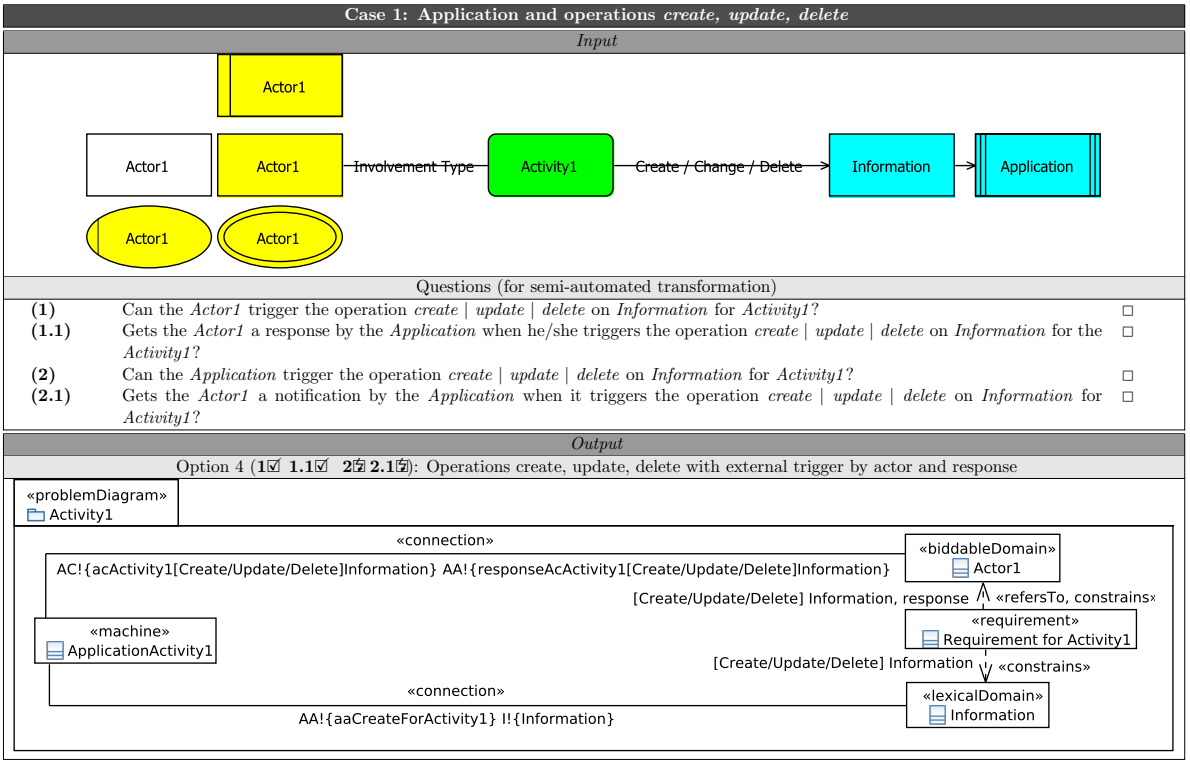


Table 12.2.: *(**Problem Diagram**) Create Phenomena: Case 1 (Option 4 out of 5 Options)*

problem diagram. Each *machine* created this way has to be *mapped* to the corresponding machine that is part of the context diagram or to one of the already known sub-machines which were aggregated to the machine in the context diagram.

For all *entities* in the FAD, we *add a lexical domain* to the problem diagram. If we add a not already known *lexical domain* this way, we search for a *mapping* of the corresponding entity to existing entities in the ERDs. If we find such a mapping, we also model the mapping in the problem frame model. In case we cannot find such a mapping, we have to create one or we have to add the new lexical domain to the related context diagram. In the same manner we *add a biddable domain* for each *position, group, organizational unit, internal person, or external person*, and *map* these *biddable domains* whenever necessary. We also *add causal domains* for the *existing applications* and *map* them if necessary.

Up to this point we added machines and domains using the entities in FAD. Next, we need to *add the connections (associations), phenomena and dependencies* between them. We distinguish five cases with different options for transformation. An example transformation table for problem diagrams is given in Table 12.2[6]. In case of a fully automated transformation, the default option is always used[7], while for semi-automated transformation answers to the questions are used to identify the transformation option.

After creating and adding the textual requirement to the problem diagram by hand, the transformation itself is finished. Now the resulting problem diagram needs to be analyzed further. First, we have to *adjust the problem diagram to be valid*. As mentioned, the combination of different transformation options might not result in a valid problem diagram. For example, it can happen that a requirement only refers to domains but no domain is constrained. Such invalid problem diagrams point out missing information, which we now have to add. Even if

[6] All transformation tables can be found in Appendix D.2 (Page 475)
[7] The default option is shown in Table D.5 (Page 475)

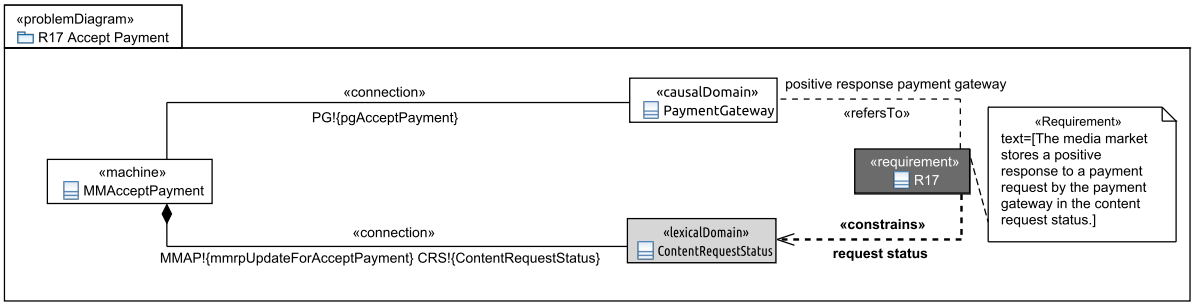


Figure 12.5.: *Problem Diagram for* `R17`*: Accept Payment*

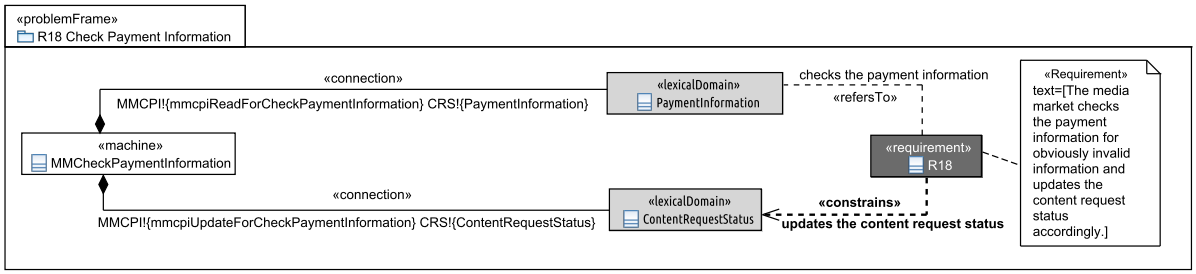


Figure 12.6.: *Problem Diagram for* `R18`*: Check Payment Information*

valid, the problem diagrams which are created from FADs often tend to be rather big. The reason is that a business activity might combine different system functions. Hence, it might be possible to *decompose a problem diagram* into smaller sub-diagrams.

The resulting problem diagrams for the FADs introduced in Section 11.2.2[8] are discussed in the following. Note that the problem diagram for the FAD *Reject Payment* (Figure 11.9[9]) was already shown and discussed in Section 2.2.4[10] (Figure 2.8[11]), while the problem diagrams for the FADs *Request Content* (Figure 11.6[12]) and *Forward Payment* (Figure 11.8[13]) are shown and discussed in Chapter 13 (Figure 13.2) and Chapter 16[14] (Figure 16.2[15]) respectively. Hence, we will only show the problem diagrams for the FADs *Accept Payment* (Figure 11.10[16]) and *Check Payment Information* (Figure 11.7[17]).

Figure 12.5 shows the problem diagram for the FAD *Accept Payment*. For fulfilling the requirement **R17** the machine **MMAcceptPayment** has to observe the phenomenon *pgAcceptPayment* issued by the causal domain **PaymentGateway**. In consequence, the machine has to *update* the **ContentRequestStatus** accordingly.

Figure 12.6 shows the problem diagram for the FAD *Check Payment Information*. In order to fulfill the related requirement **R18**, the machine **MMCheckPaymentInformation** *reads* the lexical domain **payment information**, checks it, and *updates* the lexical domain **ContentRe-**

[8] Page 209
[9] Page 209
[10] Page 35
[11] Page 37
[12] Page 209
[13] Page 209
[14] Page 281
[15] Page 283
[16] Page 209
[17] Page 209

questStatus according to the result of the check.

12.4. Conclusion

After finishing the modeling of problem diagrams, we have the functional requirements of the system-to-be in place. Within this chapter we have

- discussed what important aspects for choosing a requirements engineering method are,
- motivated the selection of the problem frames method and notation as proposed by Jackson (2001 [200]),
- shown how to use EPCs to derive a context diagram,
- shown how to use FADs to derive problem diagrams,
- presented the resulting context diagram and problem diagrams for our running example.

In particular we have contributed:

- An elaboration of criteria for selecting a suitable RE method and notation.
- A guided method for creating a context diagram and problem diagrams based on a business process model.
- This method lays the foundation for a tool support enabling the (semi-) automated transformation from business process models to problem frame models.
- In case of the semi-automated transformation additional information is created and enriches the context and problem diagrams in a lightweight way using questionnaires.
- The generated problem diagrams are a suitable input for further analysis.
- Tracing from business processes to software development artifacts is enabled.

Chapter 13

Initial Security Requirements Elicitation

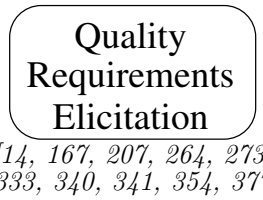


Figure 13.1.: *Quality Requirements Elicitation*

This chapter is about the step *quality requirements elicitation*, but as there are myriads of different qualities we cannot treat them all in this thesis. Hence, we have decided to concentrate on one central quality, beside legal compliance which is treated in the next step, which is of special importance for our setting: security. We will show how to break high level security goals down to security requirements which directly complement functional requirements. This chapter is based on Faßbender, Heisel, and Meis (2014 [135]), and Faßbender, Heisel, and Meis (2015 [137])[1].

Recently, there has been an increase of reported security incidents hitting large software systems. For example, in the report on cybercrime for the year 2013 published by the federal criminal police office of Germany, the authors state that 64426 security incidents were reported in Germany (Bundeskriminalamt (federal criminal police office), 2014 [85]). This is an increase by 70 percent with respect to 2008 (Bundeskriminalamt (federal criminal police office), 2013 [84]). Moreover, particular types of attacks which aim at companies increased much more. For example, data manipulation and computer sabotage incidents in companies increased by 18 percent with respect to 2012 and 578 percent with respect to 2008. These numbers are limited to Germany, but, for example, Norton reports a world wide damage of 113 billion US dollar in 2013 due to security incidents (Norton, 2013 [287]). Hence, the need for secure IT systems is staggering.

Not all of the security incidents are directly related to security defects in an IT system, but many attacks abuse indirectly or directly one or more security defects. Beside, for example, reputation damage, loss on market value and share, and costs for legal infringement (Cavusoglu, Mishra, and Raghunathan, 2004 [98]; Khansa, Cook, James, and Bruyaka, 2012 [213]), fixing the security defect causing the incident is costly. Fixing a defect when it is already fielded is reported to be up to eighty times more expensive than fixing the corresponding requirements defects early on (Willis, 1998 [391]; Boehm and Papaccio, 1988 [61]). Thus, security issues should be detected as early as possible for a system-to-be. Therefore, it is crucial for requirements engineers to identify security threats, and to refine the threats into security requirements. But eliciting good requirements is not an easy task (Firesmith, 2003 [145]), even more with regard to security, as most requirements engineers are not security experts in the first place.

In this chapter, we propose a method called problem-based security requirements elicitation

[1] The method and tool support is a contribution of the author. Also the formal specification was initially contributed by the author, but revised and beautified by Rene Meis.

(PresSuRE), which guides a requirements engineer through the process of eliciting a set of security requirements in collaboration with the stakeholders of the system-to-be and security experts. PresSuRE has several benefits. It does not require the requirements engineer to have a security background. It does not require any preliminary security requirements and security relevant information. It lowers the effort by providing tool support for semi-automated modeling and an automated security analysis. And PresSuRE is completely guided by a detailed process.

PresSuRE is based on the same idea of deriving information flows from functional requirements as the ProPAn method (Beckers, Faßbender, Heisel, and Meis, 2012 [42])[2], but changes the analysis to be suitable for security. The analysis and elicitation is based on a complete set of functional requirements for a system-to-be. The method is accompanied with tool-support[3]. PresSuRE is based on the problem frame notation.

In Section 13.1 we take closer look at a problem diagram from the running example which will serve to exemplify details of PresSuRE . The PresSuRE method is then explained in Section 13.2, and in Section 13.3 the method is validated. In Section 13.4 related work is discussed, and the final conclusion is drawn in Section 13.5.

13.1. Running Example

For showing the application of PresSuRE we will use the problem diagrams already introduced in Section 12.3 for our running example. One particular problem diagram will be of special interest as it serves as running example for showing details of the application of PresSuRE. Figure 13.2 shows the problem diagram for R3. The biddable domain **Customer** *sends a content request*, which is forwarded by the causal connection domain **MediaMarketClient** and finally observed by the machine domain **MMRequestContent**. The **MMRequestContent** machine controls the phenomenon to *create* and store the received information in the lexical domain **ContentRequest**.

Note that we will even simplify this example in the following. We will not elaborate on all security elements but restrict ourselves to one example for each element, for example, one asset, to improve the comprehensibility for the reader.

[2]ProPAn is a contribution of Rene Meis. The author only helped to improve it as well as he developed the initial tool-support for it.
[3]http://www.uml4pf.org/ext-pressure/installation.html

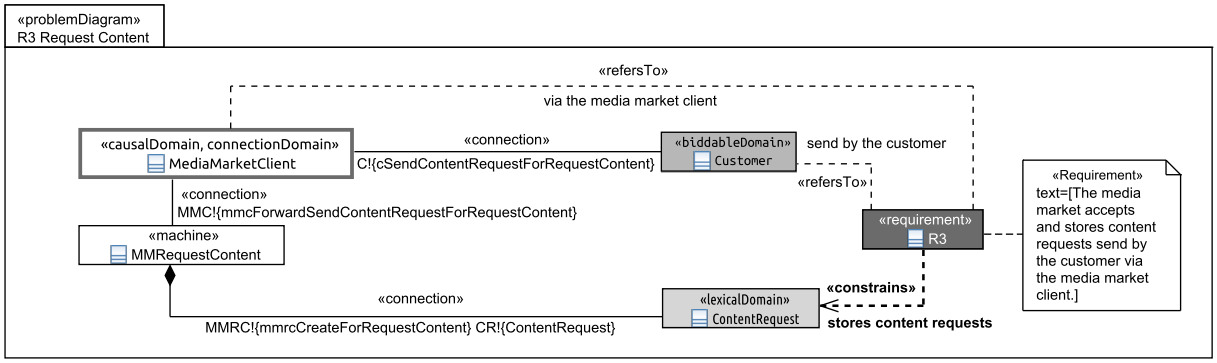


Figure 13.2.: *Problem Diagram R3: Request Content*

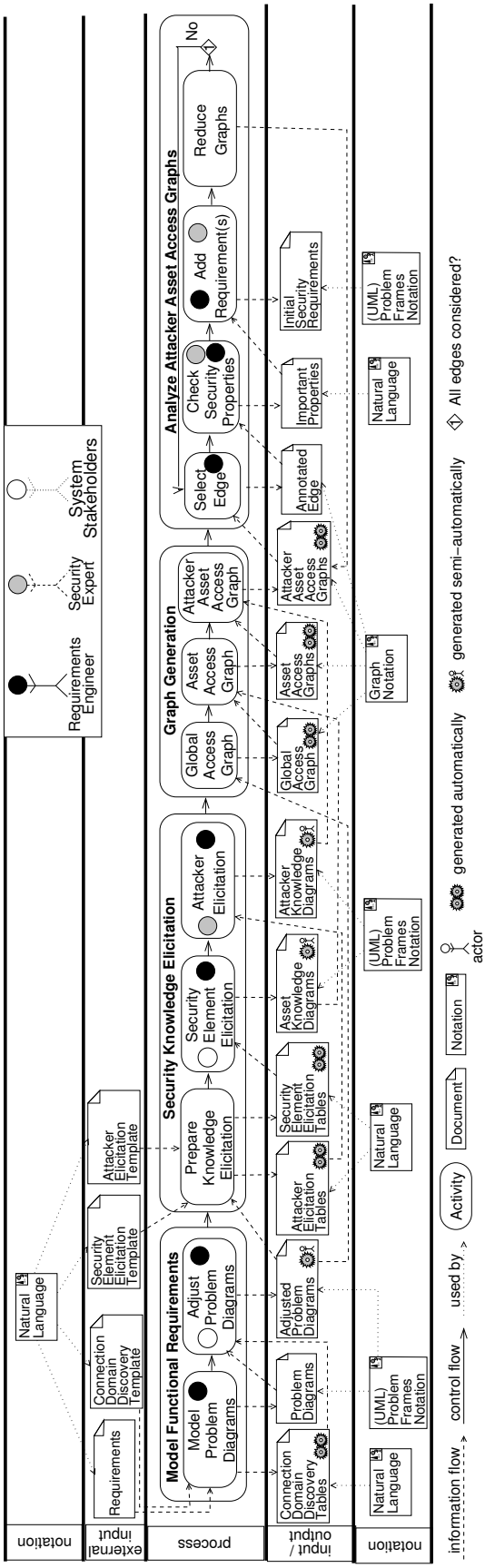


Figure 13.3.: *PresSuRE Method*

Table 13.1.: *Connection Domain Discovery Table*

Domain 1	Domain 2	phenomena	Q1+	Name?	Q2•	Name?	Q3°	Name?	PD*
...									
Customer	Media-Market-Client	C!{cSend-ContentRequest-ForRequest-Content}	No	-	No	-	No	-	R3 - Request Content
MediaMarket-Client	MM-Request-Content	MMC!{mmc-ForwardSend-ContentRequest-ForRequest-Content}	Yes	Internet	No	-	No	-	R3 - Request Content
Content-Request	MM-Request-Content	MMRC!{mmrc-CreateFor-RequestContent} CR!{Content-Request}	No	-	No	-	No	-	R3 - Request Content
...									

+Q1: Information transmitting domain involved? •Q2: Domain displaying information involved? °Q3: Domain storing information involved? *PD: Related problem diagrams

13.2. The PresSuRE Method

The PresSuRE method consists of four phases and twelve steps, which we will explain in the following (Figure 13.3 gives an overview).

13.2.1. Model functional requirements

We assume that the functionality of the system-to-be is described completely, coherently and unambiguously. The functional requirements are a good starting point for a security analysis as the requirements engineer is used to deal with them, they are often already well defined, they already contain everything which has to be protected, and they also contain the entry points for possible attack vectors an adversary can use. Note that the results PresSuRE delivers are correct and sufficient with respect to functional requirements used. Hence, having incomplete or bad requirements has an impact on the results. But how to tackle this problem is out of scope of this work. The modeling of the requirements is achieved by performing the following two steps.

Model Problem Diagrams In the first step of the PresSuRE method, the functional requirements have to be modeled using the *problem frame notation*. This can be done by the *requirements engineer* alone, based on a *textual description* of the *functional requirements*. The result is a set of *problem diagrams* as well as an automatically generated *connection domain discovery table*. *The functional requirements and corresponding problem diagrams are presented in Section 12.3, and Section 13.1*

Adjust Problem Diagrams As setting up problem diagrams allows some degree of freedom, adjustments might be needed to prepare the *problem diagrams*. For the PresSuRE analysis, connection domains are of specific importance. Many attacks use such connection domains as entry points, such as a wireless LAN for sniffing important data, a display which is visible to an attacker, or even a postman, who can be socially engineered. But as connection domains are not of central relevance for fulfilling the functional requirements, they are often left out. Hence, one has to make sure that all connection domains are explicitly modeled.

For each connection between domains, the *requirements engineer* and the *system stakeholders* have to check if there is a connection domain in between. Typical connection domains which should be modeled are domains transporting information (such as network domains or a postman), domains which show information (such as (semi)- public displays), or domains which store information for exchange between domains (such as a network-attached storage).

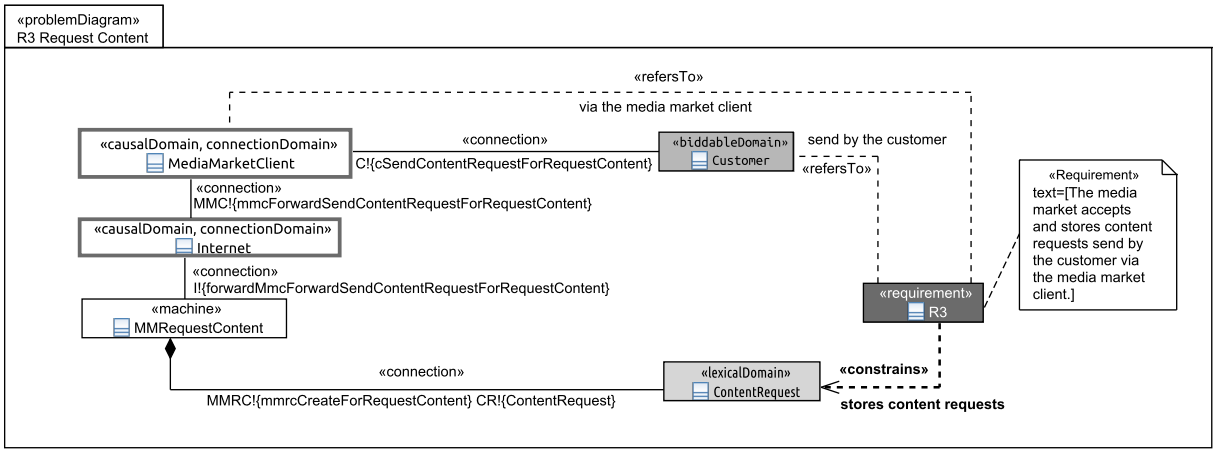


Figure 13.4.: *Adjusted Problem Diagram R3: Request Content*

While requirements engineers are able to analyze, understand, and explain the problem diagrams, and know which new connection domains might be introduced, they often lack the domain knowledge, which reveals already established connection domains. This knowledge is available through the *stakeholders of the system-to-be*. Hence, the *requirements engineer* guides through the discovery and models the results, while the *system stakeholders* provide the needed information.

The *requirements engineer* and the *system stakeholders* use a table containing the connected domains pairwise, the phenomena in between and a standard questionnaire, which helps to elicit the missing connection domains. Table 13.1 shows such a table. The first two columns show the domains which are connected, the third column the phenomena they share, the next six columns ask for the different types of connection domains and, in case there is one, for the name of the discovered domain, and the last column contains the problem diagram in which the interface at hand can be found. The result of this step are *adjusted problem diagrams*, which are modeled by the *requirements engineer* using semi-automated wizards. *For our running example, using the table and answering the questions, we discover that our the media market client is connected to the machine via the Internet. Hence, we add this information resulting in a modified problem diagram as shown in Figure 13.4. In the same manner we also discovered that the payment gateway is connected via a VPN connection. All adjusted problem diagrams can be found in Appendix F.1.5.1*[4]

13.2.2. Security Knowledge Elicitation

Before starting the security analysis, some security-specific knowledge has to be elicited. This information is crucial for the success of the analysis, as in most cases the functional requirements do not contain enough information for considering security thoroughly. The knowledge about assets in the system-to-be and attackers which might tamper with the system has to be made explicit. As this knowledge is not or only partially available for *requirements engineers*, they have to collaborate with the *stakeholders of the system* and *security experts*.

Prepare Knowledge Elicitation Even though the functional requirements do not contain all information needed for a security analysis, they do already contain some information, which is the starting point for eliciting the additional domain knowledge. We use *security element elicitation tables*, and *attacker elicitation tables* to elicit this information. The tables are automatically generated from the *problem diagrams*.

[4]Page 531

Table 13.2.: *Security Element Elicitation Table for Content Request*

Asset Candidate	is asset	Authorized Entity Candidate	is authorized entity	rights
ContentRequest	☑	PaymentGateway	□	□ read □ write
		Internet	□	□ read □ write
		MediaMarketClient	☑	☒ read ☒ write
		Accounter	☑	☒ read □ write
		ContentProvider	□	□ read □ write
		Customer	☑	☒ read ☒ write
		Administrator	□	□ read □ write
		VPN	□	□ read □ write
		ServiceBroker	□	□ read □ write

☑ : yes, ☒ : must have, ⊡ : might have

Identify assets, authorized entities and rights The baseline questions for this step are "What has to be protected?" (*asset*), "Who is eligible to access the asset?" (*authorized entities*), and "Which actions are allowed for a stakeholder regarding an asset?" (*rights*). This is in line with common security and threat analysis methods (ISO/IEC, 2009 [198]; Howard and Lipner, 2006 [189]; ISO/IEC, 2009 [199]; Jürjens, 2005 [207]). We use the previously generated *security element elicitation tables* to elicit this information. The instance for the payment data is shown in Table 13.2. Such tables are completed by the *stakeholders of the system-to-be* using the following description, while the *requirements engineer* models the results.

Assets Identify those domains, which have to be protected. Every domain beside the machine is an asset candidate. Most likely one wants to protect a lexical domain representing information or a causal domain. *For our example, we only select the content request as an asset, which contains the order information, payment data, and the request status (see Figure 11.11[5]). This information has to be protected, as it, for example, allows to monitor the customer, or payment fraud.*

Authorized Entities An authorized entity to an asset is every domain which has an eligible interest in knowing the state / reading, or controlling / writing the asset. *Authorized entities of the content request are the customer, who enters the content request, the media market client, which is used for entering the data, and the accounter, who needs this information for his / her accounting tasks. The payment gateway, which needs the contained payment data information for checking the payment data and initiating the payment is not an authorized entity for the whole request, but only for the payment data (see Table F.31[6]), because the payment gateway does not need to know anything about the actual order.*

Rights Authorized entities have different rights to access the asset. In case of a lexical domain, the rights are to read or write the information in the domain. In case of the causal domain, the rights are to control or know the state of the causal domain. For each right and authorized entity, one has to state if the entity is allowed to have the right or if the entity must have the right. *The customer as well as the media market client must have the right to read and write the information, while the accounter must have the right to read the information. The accounter is not allowed to modify the content request.*

The elicited information has to be added to the model. For this purpose, we use *domain knowledge diagrams*. In domain knowledge diagrams additional knowledge about domains and relations between domains can be modeled. To support modeling security-related domain knowledge we developed UML profiles. Figure 13.5 shows a domain knowledge diagram modeled using

[5]Page 209
[6]Page 536

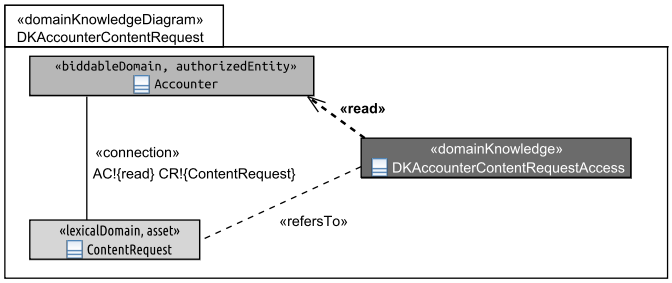


Figure 13.5.: *Asset Knowledge for the Rights of the Accounter regarding the Content Request*

the profiles. It depicts that there is domain knowledge (stereotype ≪`domainKnowledge`≫ at class *DKAccounterContentRequestAccess*) about the content request and the accounter. The content request is an asset (stereotype ≪`asset`≫ at class *ContentRequest*), the accounter is an authorized entity of this asset (stereotype ≪`authorizedEntity`≫ at class *Accounter*), and the accounter is allowed to read (≪`read`≫) from to the content request (≪`refersTo`≫). The diagrams are generated in the background while the *requirements engineer* completes a wizard which is similar to the security element elicitation table. The result of the step are *asset knowledge diagrams*[7].

Attacker(s) Elicitation In this step, the *requirements engineer* and a *security expert* have to collaborate to define those *attackers* who might attack our system-to-be. While the *requirements engineer* has a deeper understanding of the system-to-be and its domain, the *security expert* adds his/her vital knowledge about attackers, attacker abilities, possible attack vectors, and so forth. Hence, it is not mandatory that the *requirements engineer* has a security background.

Beckers, Hatebur, and Heisel (2013 [36]) enumerate different *types of attackers*: *physical attacker*, *software attacker*, *network attacker*, and *social attacker*. Regarding their abilities, we have chosen the abilities as described by Dolev and Yao (1983 [120]): *read (read message / get state of domain)*, *write (write message / change state of domain)*, *interfere (intercept message / prevent the change of state)*. Again, describing attackers with their basic abilities is in line with the state of the art (ISO/IEC, 2009 [198]; Howard and Lipner, 2006 [189]; ISO/IEC, 2009 [199]; Jürjens, 2005 [207]). For the purpose of eliciting the information about attackers, we use the generated *attacker elicitation tables* (see Table 13.3).

Attacker First, we have to reason for each domain and type of attacker about the question if this type of attacker might exist for the domain at hand. *For simplicity's sake, we assume for the running example that we only have to defend against network attackers. We also assume that the VPN connection to the gateway is perfectly secure for our purposes. Hence, we have only to deal with an Internet attacker. The Internet attacker can attack via the Internet.*

[7]Domain knowledge diagrams which contain knowledge about assets.

Table 13.3.: *Attacker Elicitation Table for Internet*

AttackCandidate	is attackable	Attacker	Name	Abilities
Internet	☑	☑ network attacker	Internet Attacker	☒ read ☒ write ☒ interfere
		□ physical attacker		□ read □ write □ interfere
		□ social attacker		□ read □ write □ interfere
		□ software attacker		□ read □ write □ interfere

☑ : yes, ☒ : has the ability

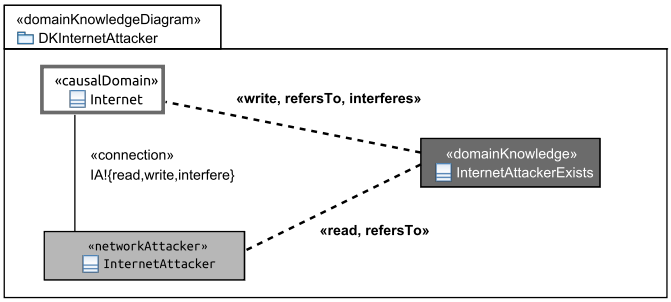


Figure 13.6.: *Attacker Knowledge regarding the Internet attacker's abilities regarding the Internet*

Abilities For each attacker and each domain the attacker has access to, we have to state which abilities the attacker has. Whenever there is no detailed information about the attackers and their abilities regarding a domain they have access to, one should assume the strongest attacker. This might lead to an overestimation of the threats afterward. But adding an unnecessary security requirement is not so much of an issue, while missing one is critical. *After an assessment of the Internet attacker and his / her abilities, we could not exclude any of the basic abilities. Hence, our Internet attacker has all abilities regarding the Internet domain he /she has access to.*

The elicited information has to be added to the model to be available for our analysis, too. Again, the modeling can be done semi-automatically using the wizards our tool provides. The result are *attacker knowledge diagrams*. Figure 13.6 shows the domain knowledge about the Internet attacker and his/her abilities regarding the Internet. It depicts that there is domain knowledge (stereotype ≪`domainKnowledge`≫ at class *InternetAttackerExists*) about the Internet attacker and the Internet. The Internet attacker is a network attacker (stereotype ≪`networkAttacker`≫ at class *InternetAttacker*). This attacker is able to read (≪`read`≫) from the Internet (≪`refersTo`≫), and the attacker (≪`refersTo`≫) is able to write to and interfere with the Internet (≪`write, interfere`≫).

13.2.3. Graph Generation

The automated part of the security analysis relies on graphs, which visualize information flows and access flows. The attacker asset access graphs, which contain the potential security threats towards the functional requirements, are generated stepwise. The steps and intermediate graphs are explained in the following.

Global Access Graph All graphs $(\mathcal{V}, \mathcal{E})$ that we use for our security analysis in the PresSuRE method are labeled and directed. The set of vertices is a subset of the domains occurring in the model, formally $\mathcal{V} \subseteq Domain$. An edge is annotated with a diagram and a type. The diagram can be a problem diagram or a domain knowledge diagram. The type can be required (*req*), implicit (*imp*) or attack (*att*) ($Type ::= \text{req}|\text{imp}|\text{att}$). The type indicates if the edge is *required* or *implicitly* given by the problem diagram or if it shows a possible *attack* relationship defined in a domain knowledge diagram. The edges point from one domain to another, formally $\mathcal{E} \subseteq Domain \times Diagram \times Type \times Domain$. For the rest of the chapter we will regard such an edge as an access flow. In the following, we describe a graph $(\mathcal{V}, \mathcal{E})$ only by its edges $\mathcal{E}$.

For the analysis of the threats towards an asset we will use the *global access graph*. This graph contains the information about access flows between domains, and which problem diagrams are the source of these flows. For the flows, we distinguish between required flows as stated by the requirement and implicit ones which are modeled due to the given environment. To set up the global access graph we use the problem diagrams as an input. The predicates $constrains, refersTo : (Domain \times Diagram)$ and $controls : \mathbb{P}(Domain \times Domain \times Diagram)$

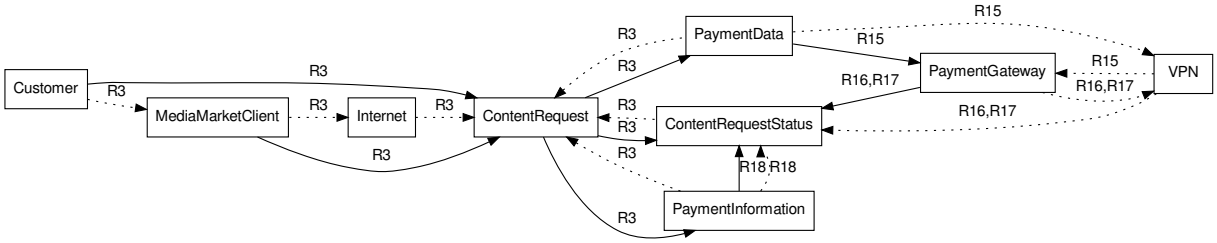


Figure 13.7.: *Global Access Graph (also Asset Access Graph for Content Request)*

can be derived from the problem frame model and are used to generate the global access graph. We have $(d, p) \in constrains$ and $(d, p) \in refersTo$ iff a requirement or domain knowledge in diagram p constrains the domain d or refers to it, respectively. $(d_1, d_2, p) \in controls$ is true iff the domain d_1 controls an interface that d_2 observes in the diagram p.

Using these predicates, we create the global access graph $\mathcal{G}$, which is an overapproximation of the access flows occurring in the system-to-be. An edge $(d_1, p, \text{req}, d_2)$ is in $\mathcal{G}$ iff the domains d_1 and d_2 are not equal, and the domain d_1 is referred to and the domain d_2 is constrained in p. *For example, the problem diagram for* `R3` *(see Figure 13.4) contains the customer and the content request. The customer is referred by* `R3` *and the content request is constrained by* `R3`*. Hence, we add a required access flow edge (solid arrow) between customer (node with name Customer) and content request (node with name ContentRequest) annotated with* `R3` *(see graph shown in Figure 13.7).*

Additionally, an edge $(d_1, p, \text{imp}, d_2)$ is in $\mathcal{G}$ iff $(d_1, p, \text{req}, d_2)$ is not already in $\mathcal{G}$, the domains d_1 and d_2 are not equal, and d_2 observes an interface controlled by d_1 in p. Note that machines are treated as transitive forwarders in this case. This means that whenever a machine m observes an interface controlled by d_1 and d_2 observes an interface controlled by m, we assume that d_2 observes an interface of d_1. *For example, the domain Internet controls a phenomenon for forwarding the content request which is observed by the machine (see Figure 13.4). The domain content request observes a phenomenon from the machine. Hence, an implicit access flow edge (dotted edge) is added between the Internet and the content request annotated with* `R3` *(see Figure 13.7).* We define the graphs as follows.

$$
\begin{aligned}
\mathcal{G}_{req} =& \{(d_1, p, t, d_2) : Domain \times ProblemDiagram \times \{\text{req}\} \times Domain \mid \\
& d_1 \neq d_2 \wedge (d_2, p) \in constrains \wedge (d_1, p) \in refersTo\} \\
\mathcal{G}_{imp} =& \{(d_1, p, t, d_2) : Domain \times ProblemDiagram \times \{\text{imp}\} \times Domain \mid \\
& d_1 \neq d_2 \wedge (d_1, p, \text{req}, d_2) \notin \mathcal{G}_{req} \wedge ((d_1, d_2, p) \in controls \\
& \vee \exists m : Machine \bullet (d_1, m, p) \in controls \wedge (m, d_2, p) \in controls)\} \\
\mathcal{G} =& \mathcal{G}_{req} \cup \mathcal{G}_{imp}
\end{aligned}
$$

Because of the annotation of the edges we keep the information which problem diagram causes the access flow. Thus, our global access graph contains traceability links that are used in our further analysis. The semantics of an edge $(d_1, p, t, d_2) \in \mathcal{G}$ is that in problem diagram p there is possibly a required or implicit (depending on t) access flow from domain d_1 to domain d_2.

Asset Access Graph As the global access graph can be huge for a complex system-to-be, we introduce an asset access graph which focuses the view on one asset only. It only contains access flows given by the requirements directly or indirectly concerning the asset. Thus, we get one asset access graph per asset. The asset access graph makes the information for the requirements engineer easier to comprehend. Hence, it improves the scalability of our method. An edge (d_1, p, t, d_2) is in $\mathcal{G}_{asset}$ iff p is in $\mathcal{P}_{access}$. A problem diagram p is in $\mathcal{P}_{access}$ iff there is an

edge (d_1, p, t, d_2) which is required and d_1 and d_2 are both in $\mathcal{D}_{access}$. $\mathcal{D}_{access}$ is a union of $\mathcal{D}_{active}$ and $\mathcal{D}_{passive}$. A domain d_1 is in $\mathcal{D}_{active}$ iff there is a required access flow which starts at d_1 and the target domain d_2 is already in $\mathcal{D}_{active}$. Initially, only the *asset* is in $\mathcal{D}_{active}$. Hence, $\mathcal{D}_{active}$ contains all domains which have a required direct or indirect (via another domain) access flow towards the asset. A domain d_2 is in $\mathcal{D}_{passive}$ iff there is a required access flow which ends at d_2 and the source domain d_1 is already in $\mathcal{D}_{passive}$. Initially, only the *asset* is in $\mathcal{D}_{passive}$. Hence, $\mathcal{D}_{passive}$ contains all domains which are the target of a required direct or indirect (via another domain) access flow from the asset. Formally, we define $\mathcal{D}_{active}$, $\mathcal{D}_{passive}$, $\mathcal{D}_{access}$, $\mathcal{P}_{access}$, and $\mathcal{G}_{asset}$ as follows.

$$
\begin{aligned}
&\textbf{required } asset \in Domain\\
&asset \in \mathcal{D}_{active}, \quad asset \in \mathcal{D}_{passive},\\
&(d_1, p, \text{req}, d_2) \in \mathcal{G} \land d_2 \in \mathcal{D}_{active} \Rightarrow d_1 \in \mathcal{D}_{active}\\
&(d_1, p, \text{req}, d_2) \in \mathcal{G} \land d_1 \in \mathcal{D}_{passive} \Rightarrow d_2 \in \mathcal{D}_{passive}\\
&\mathcal{D}_{access} = \mathcal{D}_{active} \cup \mathcal{D}_{passive}\\
&\mathcal{P}_{access} = \{p : ProblemDiagram \mid \exists (d_1, p, \text{req}, d_2) : \mathcal{G} \bullet d_1, d_2 \in \mathcal{D}_{access}\}\\
&\mathcal{G}_{asset} = \{(d_1, p, t, d_2) : \mathcal{G} \mid p \in \mathcal{P}_{access}\}
\end{aligned}
$$

The resulting asset access graph for the content request is shown in Figure 13.7, as for our small example the global and the asset access graph do not differ. For a complex scenario the asset access graph is significantly smaller than the global access graph. The asset access graph can be used to check if a stakeholder can gain more rights than he/she should. For reasons of space, we do not go into detail on this matter.

Attacker Asset Access Graph For each asset, we generate the attacker asset access graph, which visualizes the information and control flows from attackers to the asset and from the asset to the attackers. At this point, we focus on the basic information security goals confidentiality, integrity, and availability (short CIA), which are suggested by the Common Criteria (ISO/IEC, 2009 [198]) and ISO 27000 family of standards (ISO/IEC, 2009 [199]). The problematic access flows are annotated with the information which CIA property(ies) are threatened ($CIA ::= \text{C}|\text{I}|\text{A}|\varepsilon$). First, the domains which are directly connected to attackers are identified. Note that for this purpose we use the information given in domain knowledge diagrams created during the step *Identify assets, authorized entities and rights* described in Section 13.2.2. From these diagrams, we can derive the predicates $read, write, interfere : \mathbb{P}(Domain \times Diagram)$. We have $(d, dk) \in read$, $(d, dk) \in write$, and $(d, dk) \in interfere$ iff domain knowledge in diagram dk has a read, write, or interfere dependency, respectively, to the domain d.

A domain d can be subject of an attack if it is in $\mathcal{D}_{access}$ for the asset at hand. That is, an attacker can access or influence information on the asset through the domain d. We define the sets $\mathcal{D}_w$, $\mathcal{D}_i$, and $\mathcal{D}_r$ as the sets of all domains for which an attacker has the ability to *write*, *interfere*, or *read* it, respectively. A domain d is in $\mathcal{D}_w$ iff there exists an attacker a and a domain knowledge diagram dk, in which d is written and a is referred to by the domain knowledge. The domain d is in $\mathcal{D}_i$ iff there exists an attacker a and a domain knowledge diagram dk in which d is interfered with and a is referred as source of the interference. The domain d is in $\mathcal{D}_r$ iff there exists an attacker a and a domain knowledge diagram dk in which the information in d is referred to and a reads this information. Based on the three sets of domains which might be attacked, the asset threat graph $\mathcal{G}_{threat}$ can be set up. $\mathcal{D}_w$, $\mathcal{D}_i$, and $\mathcal{D}_r$ are formally defined as follows.

$$
\begin{aligned}
\mathcal{D}_w &= \{d : \mathcal{D}_{access} \mid \exists a : Attacker; dk : Diagram \bullet (d, dk) \in write \land (a, dk) \in refersTo\}\\
\mathcal{D}_i &= \{d : \mathcal{D}_{access} \mid \exists a : Attacker; dk : Diagram \bullet (d, dk) \in interfere \land (a, dk) \in refersTo\}
\end{aligned}
$$

$$\mathcal{D}_r = \{d : \mathcal{D}_{access} \mid \exists a : Attacker; dk : Diagram \bullet (a, dk) \in read \wedge (d, dk) \in refersTo\}$$

$\mathcal{G}_{threat}$ contains all edges, and therefore problem diagrams, of the corresponding asset access graph which might allow an attacker to successfully attack the asset at hand. An access flow $(d_1, p, t, d_2) \in \mathcal{G}_{asset}$ represents that information is transferred from d_1 to d_2 that possibly comes from the asset or that possibly will be stored in the asset. Hence, such an access flow is a possible threat to the confidentiality of an asset if an attacker has the ability to read one of the domains d_1 or d_2 ($d_1 \in \mathcal{D}_r \vee d_2 \in \mathcal{D}_r$). In this case, we add the edge $(d_1, p, t, \mathrm{C}, d_2)$ to $\mathcal{G}_{threat}$. An access flow $(d_1, p, t, d_2) \in \mathcal{G}_{asset}$ is a possible threat to the integrity of an asset if an attacker has the ability to write the source d_1 of the access flow ($d_1 \in \mathcal{D}_w$), because an attacker could change the information of the asset or the information sent to the asset at domain d_1, which forwards it to domain d_2. In this case, we add the edge $(d_1, p, t, \mathrm{I}, d_2)$ to $\mathcal{G}_{threat}$. We have to consider an access flow $(d_1, p, t, d_2) \in \mathcal{G}_{asset}$ as a possible threat to the availability of an asset if an attacker has the ability to interfere with one of the domains d_1 or d_2 ($d_1 \in \mathcal{D}_i \vee d_2 \in \mathcal{D}_i$), because an attacker is then able to threaten the availability of information flowing from or to the asset through the domains d_1 and d_2. In this case, we add the edge $(d_1, p, t, \mathrm{A}, d_2)$ to $\mathcal{G}_{threat}$. $\mathcal{G}_{threat}$ is defined as follows.

$$\begin{aligned}\mathcal{G}_{threat} = \{&(d_1, p, t, cia, d_2) : Domain \times Diagram \times Type \times CIA \times Domain \mid \\ &(d_1, p, t, d_2) \in \mathcal{G}_{asset} \wedge [(d_1 \in \mathcal{D}_r \vee d_2 \in \mathcal{D}_r) \wedge cia = \mathrm{C} \vee d_1 \in \mathcal{D}_w \wedge cia = \mathrm{I} \\ &\vee (d_1 \in \mathcal{D}_i \vee d_2 \in \mathcal{D}_i) \wedge cia = \mathrm{A}]\}\end{aligned}$$

The full attacker asset access graph $\mathcal{G}_{attack}$ is an extension of $\mathcal{G}_{threat} \subset \mathcal{G}_{attack}$. We add an edge $(d_1, p, t, \varepsilon, d_2)$ to $\mathcal{G}_{attack}$ iff (d_1, dk, t, d_2) is in $\mathcal{G}_{asset}$ but not in $\mathcal{G}_{threat}$. These edges visualize how the attacks on the access flows in $\mathcal{G}_{threat}$ might be propagated over the system due to the functional requirements. Additionally, the attackers are added to the attacker asset access graph. $\mathcal{G}_{attack}$ contains an edge $(a, dk, \mathrm{att}, cia, d)$ if a is an attacker and a domain knowledge diagram dk exists, in which d is referred to and d is written ($cia = \mathrm{I}$) or interfered ($cia = \mathrm{A}$). Additionally, $\mathcal{G}_{attack}$ contains an edge $(a, dk, \mathrm{att}, \mathrm{C}, d)$ if a is an attacker and a domain knowledge diagram dk exists in which d is referred to and a is read. Formally, we define $\mathcal{G}_{attack}$ as follows.

$$\begin{aligned}\mathcal{G}_{attack} = \{&(d_1, p, t, \varepsilon, d_2) : Domain \times Diagram \times Type \times CIA \times Domain \mid \\ &(d_1, p, t, d_2) \in \mathcal{G}_{asset} \wedge \forall st : CIA \bullet (d_1, p, t, st, d_2) \notin \mathcal{G}_{threat}\} \cup \\ &\{(a, dk, \mathrm{att}, cia, d) : Attacker \times Diagram \times Type \times CIA \times Domain \mid \\ &(d, dk) \in refersTo \wedge [(a, dk) \in read \wedge cia = \mathrm{C} \vee (a, dk) \in write \wedge cia = \mathrm{I} \\ &\vee (a, dk) \in interfere \wedge cia = \mathrm{A}]\} \cup \mathcal{G}_{threat}\end{aligned}$$

The generated attacker asset access graph for the content request is shown in Figure 13.8. Note

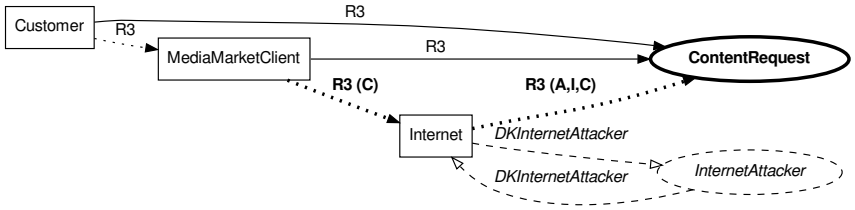


Figure 13.8.: *Attacker Asset Access Graph for Content Request*

that for reasons of readability, the PresSuRE tool merges edges and their annotation if they have the same source and target, and are of the same type. The *asset* is now visualized as *ellipse with bold border* and the asset name (*ContentRequest*) is written in bold. The *attacker* Internet attacker is also added as an *ellipse with dashed borders* and in italic font. His / her *attack flow edges* are shown as dashed edges, which are annotated with the domain knowledge diagram they are described in and the security goals they may threaten. A bold (both, edge and annotation) access flow indicates a flow for which a security property might be threatened by an attacker. The threatened security property is annotated in brackets. For example, the implicit access flow edge between the nodes *Internet* and *ContentRequest* is annotated with *R3 (A,C,I)*. Hence, it might be possible that for R3 the confidentiality, integrity and availability of the content request is threatened.

13.2.4. Analyze Attacker Asset Access Graph

For the last phase of PresSuRE we have to *analyze the attacker asset access graphs* and derive initial security requirements. The input to this step are the attacker asset access graphs. The attacker asset access graph contains all information regarding access flows to and from the asset at hand. And it contains the information where the asset might be threatened by an attacker. For each asset we identified previously, we check if we have to augment the original requirements related to the asset with security requirements. For each attacker asset access graph, we have to do the following as long as not all problematic access flows are treated:

Select edge Select a problematic required or implicit access flow (bold edge with bold annotation) not considered yet. *We select the implicit access flow edge between the nodes Internet and ContentRequest annotated with RQ3 (A,C,I)*

Check confidentiality If there is a (. . . , C, . . .) annotated, we have to check whether there is a threat to the confidentiality of the asset or not. If the threat can occur for the annotated requirement, we have to augment this requirement with a confidentiality requirement. *Indeed, the confidentiality is threatened by an Internet attacker. If he / she is able to learn all data sent by the customers, he / she learns important information for, for example, payment fraud. Hence, we have to add a confidentiality requirement complementing R3.*

Check integrity If there is an (. . . , I, . . .) annotated, we have to check whether there is a threat to the integrity of the asset or not. If the threat can occur for the annotated requirement, we have to augment this requirement with an integrity requirement. *The integrity is threatened by an Internet attacker. If he / she is able to add data or change data sent by the customers, he / she can alter the complete request and cause, for example, financial damage this way. Another attack might be to compromise the customer by requesting media which are not allowed in his / her country. Hence, we have to add an integrity requirement complementing R3.*

Check availability If there is an (. . . , A, . . .) annotated, we have to check whether there is a threat to the availability of the asset or not. If the threat can occur for the annotated requirement, we have to augment this requirement with an availability requirement. *The availability can be threatened by an Internet attacker. If he / she is able to deny the service of the Internet, no data can be sent by the customers. Of course it is not possible to make the Internet completely dysfunctional, but in case, for example, a man in the middle or a denial of service attack, an attacker might be able to block the access of the customer. Hence, we have to add an availability requirement complementing R3.*

The iteration over the assets, and the iteration over the edges in an according attacker asset access graph for the asset at hand, is guided by the tool. It indicates the asset and the edge in question and shows the according attacker asset access graph. The requirements engineer and security expert have to do the reasoning and provide the result to the tool. From this information we collected for an edge, we can derive initial security requirements. The initial

Input
domain accessed by the attacker, *the attacker*, and *threatened security property*
Precondition
Precondition 1: Domain accessed by the attacker is a connection domain
Precondition 2: Security property threatened is integrity or confidentiality
Template
Title: Secure access flows via [*domain accessed by the attacker*]
Text: The access flows via the [*domain accessed by the attacker*] must be secured in a way such that the [*the attacker*] is not able to threaten the [*threatened security property*] of the access flows.

Table 13.4.: *SR Template for Connection Domains, and Integrity and Confidentiality*

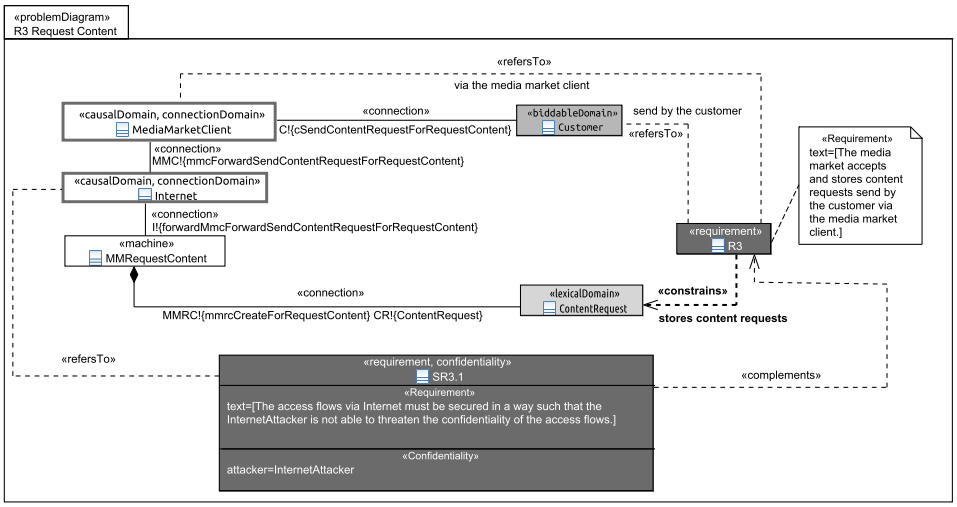


Figure 13.9.: *Problem Diagram for* `R3` *Augmented with SRQ3.1*

security requirements can be generated automatically, by using templates. For example, the template for a security requirement regarding a connection domain which can be accessed by an attacker to threaten the security properties confidentiality and integrity is shown in Table 13.4. Such a template defines the inputs for filling the templates. In this case, we need the attacker, the domain he/she can access and the security property threatened by the access of the attacker. To instantiate the template in a reasonable way some preconditions must be fulfilled. First, the domain accessed by the attacker must be a connection domain. Second, the security property threatened must be integrity or confidentiality. The template itself is given as gap-text in which the gaps are indicated by brackets. Within a bracket the input element is referenced, which will later on replace the bracket when instantiating the template. Such a template also contains the modeling rules to add the security requirement to the problem frames model. For sake of brevity, we do not show and discuss these rules in detail. An example model is shown in Fig. 13.9. In general, we stick to the profile and rules as defined in (Hatebur and Heisel, 2010 [174]).

The templates for the different cases are implemented in the tool. *Hence, for our example we can generate the following requirement regarding confidentiality:* ***SR 3.1 Secure access flows via*** **Internet** *The access flows via the* Internet *must be secured in a way that the* InternetAttacker *is not able to threaten the* confidentiality *of the access flows. Figure 13.9 shows the according modeling, in which the confidentiality requirement SR3.1 (UML class with stereotypes ≪*`requirement, confidentiality`*≫) complements (UML dependency with stereotype ≪*`complements`*≫) R3 (UML class with stereotype ≪*`requirement`*≫). SR3.1 constrains (UML dependency with stereotype ≪*`constrains`*≫) the Internet (UML class with stereotypes ≪*`connectionDomain, causalDomain`*≫). SR3.1 considers the InternetAttacker (Property at-*

tackers of SR3.1). We treat the integrity and availability threat for the selected edge in the same way.

Every newly added security requirement has an impact on the attacker asset access graph at hand. But it also has an impact on other attacker asset access graphs whenever an attacker asset access graph contains edges, which appear due to the functional requirement that is complemented by the newly added security requirement. Hence, it is necessary to reduce all attacker asset access graphs to ensure that one only analyzes edges which are not already treated by a security requirement. For the specification of the reduction of attacker asset access graphs, we need two additional predicates. The predicate $isMitigated : \mathbb{P}(Requirement \times CIA \times Attacker)$ can be derived from the problem frame model. We have $(r, cia, a) \in isMitigated$ iff the requirement r is *complemented* by a *cia* security requirement, which refers to attacker a. The predicate $models : \mathbb{P}(Requirement \times Diagram)$ can be derived from the problem frame model. We have $(r, p) \in models$ iff a requirement r is part of the diagram p. Additionally, we define the set $\mathcal{R}_{access}$. $\mathcal{R}_{access}$ contains the tuples $(r, cia, a) : Requirement \times CIA \times Attacker$ which relate the requirement r to the attacker a who exploits r to threaten the security property *cia*. A tuple (r, cia, a) is in $\mathcal{R}_{access}$ iff an access flow (d_1, p, t, cia, d_2) exists in part of the attacker asset access graph G_{attack} for which the requirement r is modeled in the diagram p and $\mathcal{G}_{attack}$ additionally contains an edge $(a, dk, \text{att}, cia, d_1)$ or an edge $(a, dk, \text{att}, cia, d_2)$.

$$\begin{aligned}\mathcal{R}_{access} =& \{(r, cia, a) : Requirement \times CIA \times Attacker \mid \exists (d_1, p, t, cia', d_2) : \mathcal{G}_{attack} \bullet \\ & cia' = cia \wedge (r, p) \in models \wedge (\exists dk : diagram \bullet (a, dk, \text{att}, cia, d_1) \in \mathcal{G}_{attack} \\ & \vee (a, dk, \text{att}, cia, d_2) \in \mathcal{G}_{attack}\}\end{aligned}$$

Based on $\mathcal{R}_{access}$ and $\mathcal{G}_{threatOld}$, which is equal to $\mathcal{G}_{threat}$ calculated before we introduce a new security requirement, we can now update $\mathcal{G}_{threat}$. $\mathcal{G}_{threat}$ now contains all edges, and therefore problem diagrams, of the corresponding asset access graph which might allow an attacker to successfully attack the asset at hand and this attack is not mitigated by an according security requirement. An access flow $(d_1, p, t, cia, d_2) \in \mathcal{G}_{threatOld}$ is also contained in $\mathcal{G}_{threat}$ iff there exists an requirement r and an attacker a for which the requirement r is modeled in the diagram p, the requirement r enables the attacker a to threaten *cia*, and this access is still not mitigated by a complementing security requirement. Formally, we define the new $\mathcal{G}_{threat}$ as follows:

required $\mathcal{G}_{threatOld}$

$$\begin{aligned}\mathcal{G}_{threat} =& \{(d_1, p, t, cia, d_2) : \mathcal{G}_{threatOld} \mid (\exists r : Requirement, a : Attacker \bullet \\ & (r, p) \in models \wedge (r, cia, a) \in \mathcal{R}_{access} \setminus isMitigated)\}\end{aligned}$$

The updated threat graph $\mathcal{G}_{threat}$ leads to an updated and reduced attacker asset access graph $\mathcal{G}_{attack}$. Hence, the tool ensures that only edges are analyzed which are not already treated. Additionally, the tool is now able to detect that an asset is not threatened anymore as $\mathcal{G}_{attack}$ is gradually reduced until it is empty. *After we have added a security requirement for confidentiality complementing R3, the tool generates reduced attacker asset access graphs. Figure 13.10 shows the graph for the content request after the reduction (the initial graph is shown in Fig 13.8). The implicit access flow edge between MediaMarketClient and Internet is not threatened any longer, and the implicit access flow edge between Internet and Content Request is only threatened with*

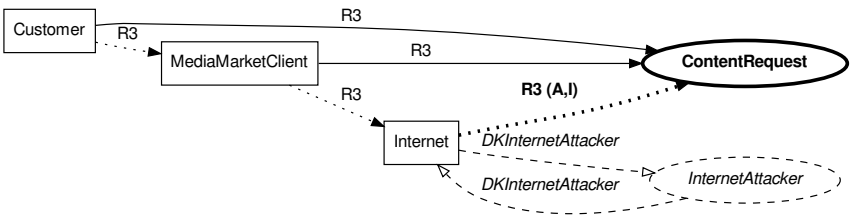


Figure 13.10.: *Attacker Asset Access Graph for Content Request After Reduction*

regards to integrity and availability. After adding integrity and availability requirements, the attacker asset access graph is empty.

13.3. Validation

We validated PresSuRE using two real-life cases, the already introduced smart meter and voting system. The results for applying PresSuRE are reported in the following in detail for the Smart Grid. For conducting our method, we selected 13 minimum uses cases, which embody 27 requirements in total. For these requirements, 14 assets and 7 attackers of all kinds, as described in Section 13.2.2, were identified. Based on this information, the graphs were generated, and the initial security requirements elicited.

We analyzed each attacker asset access graph for assessing the tool support, and we also analyzed the initial security requirements found for assessing the overall method. For the graph, we checked for each edge in the attacker asset access graph at hand if the annotated threats are existing according to the threats and security requirements of the original documents (for example, (OPEN meter project, 2009 [297]; BSI, 2011 [76]) for smart meter). We also looked for threats and security requirements which are defined in such documents, but which were not identified using PresSuRE. In this way, we were able to measure the precision and recall of our method. Unfortunately, we do not know which security analysis was used for eliciting the security requirements reported in those documents. But we assume that security experts were involved in writing the documents and the documents were reviewed thoroughly. Hence, these documents are a good benchmark.

Next, we aggregated the results of the edges of the attacker asset access graph for each requirement. Thus, we derived for each requirement the information if the requirement has to be complemented by security requirements according to PresSuRE. Again, we also checked if the found security requirements are compliant with the original documents. Last, we measured the precision and recall of PresSuRE on the requirements level.

The results of this analysis for the smart meter are shown in Table 13.5. Speaking of the precision on the level of edges of the attacker asset access graph, we have many false positives, especially for confidentiality. This is because the original documents do not demand a high level of confidentiality. Additionally, PresSuRE discovered potential indirect information flows between assets which will not occur in the system later on. Thus, PresSuRE is very strict and defensive, which is not appropriate in every case. Note that even though the indirect flows often turned out to be irrelevant, they have to be checked anyway. Often, attacks use such indirect relations to tamper with a system. Overall, the precision on the level of edges of the attacker asset access graph is acceptable (55%), but should be improved. The recall is perfect (100%) as we did not find any false negatives. On the requirements level, our results are satisfying. Whenever PresSuRE suggested to add a complementing security requirement for a functional

Table 13.5.: *Results of the assessment for the smart meter case study*

		Confidentiality	Integrity	Availability
Smart Meter	Precision per attack asset access edges	36.14%	66.35%	62.52%
	Recall per attack asset access edges	100.00%	100.00%	100.00%
	Aggregated precision per attack asset access edges	55.00%		
	Aggregated recall per attack asset access edges	100.00%		
	Precision per requirement	92.59%	96.30%	96.30%
	Recall per requirement	100.00%	100.00%	100.00%
	Aggregated precision per Requirement	95.06%		
	Recall per requirement	100.00%		
Voting system	Precision per attack asset access edges	89.74%	94.02%	94.02%
	Recall per attack asset access edges	100.00%	100.00%	100.00%
	Aggregated precision per attack asset access edges	92.60%		
	Aggregated recall per attack asset access edges	100.00%		
	Precision per requirement	95.00%	95.00%	95.00%
	Recall per requirement	100.00%	100.00%	100.00%
	Aggregated precision per Requirement	95.00%		
	Recall per requirement	100.00%		

Table 13.6.: *Effort spent for conducting the method*

		Method Step								
		Model Functional Requirements		Security Knowledge Elicitation			Graph Generation			Analyze Attacker Access Graphs
		Model Problem Diagrams	Adjust Problem Diagrams	Prepare Knowledge Elicitation	Security Element Elicitation	Attacker(s) Elicitation	Global Access Graph	Asset Access Graph	Attacker Asset Access Graph	
Effort	Smart Meter									
	Ø Per Item	**28** min. per *Problem Diagram*	**5.5** min. per *Domain*	**< 1** sec. per *Domain*	**10** min. per *Domain*	**12** min. per *Domain*	**3** sec. per *Problem Diagram*	**9** sec. per *Asset*	**40** sec. per *Attacker* and *Asset*	**15** min. per *Problem Diagram*
	Number of Items	**27** *Problem Diagrams*	**17** *Domains*	**20** *Domains*	**20** *Domains*	**20** *Domains*	**27** *Problem Diagrams*	14 *Assets*	**14** *Assets* **7** *Attackers*	**27** *Problem Diagrams*
	Total	**12.5** person hours	**1.6** person hours	**0** computer hours	**3.3** person hours	**4** person hours	**0** computer hours	**0** computer hours	**1** computer hours	**6.75** person hours
	# Involved Persons	1	2	0	2	2	0	0	0	2
	Accumulated Total	**12.5** person hours	**3.2** person hours	**0** computer hours	**6.6** person hours	**8** person hours	**0** computer hours	**0** computer hours	**1** computer hours	**13.5** person hours
	Voting System									
	Ø Per Item	**37** min. per *Problem Diagram*	**10** min. per *Domain*	**< 1** sec. per *Domain*	**11** min. per *Domain*	**10** min. per *Domain*	**13** sec. per *Problem Diagram*	**44** sec. per *Asset*	**235** sec. per *Attacker* and *Asset*	**23** min. per *Problem Diagram*
	Number of Items	**20** *Problem Diagrams*	**10** *Domains*	**15** *Domains*	**15** *Domains*	**15** *Domains*	**20** *Problem Diagrams*	7 *Assets*	**7** *Assets* **2** *Attackers*	**20** *Problem Diagrams*
	Total	**12.3** person hours	**1.6** person hours	**0** computer hours	**2.75** person hours	**2.5** person hours	**0** computer hours	**0** computer hours	**1** computer hours	**7.6** person hours
	# Involved Persons	1	1	0	2	2	0	0	0	1
	Accumulated Total	**12.3** person hours	**1.6** person hours	**0** computer hours	**4.5** person hours	**5** person hours	**0** computer hours	**0** computer hours	**1** computer hours	**7.6** person hours

requirement, this suggestion was correct with a precision of 95%, and no security requirement was missed (recall 100%).

Similar results were obtained for the voting system case regarding the requirements level. The precision on the level of edges of the attacker asset graph is noticeably higher, as the voting system documents are very strict regarding all security properties. Hence, we have an indication that the more security is considered to be the central concern, the better PresSuRE performs as it is very strict regarding the security properties. For the attacker asset access edge and the requirements level the recall was 100% again.

Speaking of the effort (see Table 13.6), the 43 person hours spent are a significant effort, but seem to be reasonable. The main effort is spent on the domain knowledge elicitation and the security analysis. Both are vital for security requirements elicitation and have to be conducted even when not using PresSuRE. But PresSuRE speeds up these activities and lowers the error rate, as it provides guidance and means for a detailed analysis and discussion. Additionally, from our experience, PresSuRE scales well, as most steps rely on the complexity of the environment. And the environment does not grow significantly when adding further requirements. When discussing the requirements itself, PresSuRE enables to focus on local and small problems, which also supports the scalability in means of comprehensibility and an only linear increase in time for the security analysis.

For the voting system, we observed no significant difference in comparison with the smart meter system regarding the effort spent. The only observation which might be noticeable is that the generation of graphs does not only rely on the number of problem diagrams, but also to which extent the requirements are related to each other. For the voting system we have only 20 problem diagrams compared to 27 for the smart meter, but still the generation of graphs is slower, because for the voting system all requirements are closely related to each other, while for the smart meter there are clusters of requirements and those clusters are hardly related. But in the end, this has no real impact on the effort, as the graph generation is done automatically and the time needed for generating the graphs is quite low.

13.4. Related Work

Schmidt and Jürjens (2011 [340]) propose to integrate the SEPP method, which is based on problem frames, and UMLSec [207], which is based on a UML profile and allows tool-based reasoning about security properties. In this way, they can express and refine security require-

ments and transfer the security requirements to subsequent design artifacts. A similar method is described by Haley et al. (2008 [167]), which also relies on problem frames for security requirements analysis. The first method (Schmidt and Jürjens, 2011 [340]) starts after the initial security requirements are already known, while the latter one already embodies a step for security requirements elicitation. But this particular step is described very sparsely and informally. Hence, our work can complement and improve these works. In general, the SEPP method by Schmidt (2010 [341]) is a promising follow up to our method.

There are many publications concerning goal-oriented security requirements analysis (for example, Mouratidis and Giorgini (2007 [273]), Salehie et al. (2012 [333]), and Van Lamsweerde (2004 [377])). But as already discussed, goal models are of a higher level of abstraction than problem frames. Goal models are stakeholder-centric, while problem frames are system-centric. Therefore, refining functional requirements taking into account more detail of the system-to-be and analyzing the system-to-be described by the functional requirements is reported to be difficult for goal-oriented methods (Alrajeh et al., 2009 [14]). Alrajeh et al. (2009 [14]) try to tackle this problem by introducing refinement steps which rely on heavyweight formalizations. We offer an alternative way of bridging the this gap. Thus, even though the goals of an attacker and their implication for the goals of stakeholders are already known, one might benefit from using our method, because our method enables one to break down the actual security goals to specific security requirements. Additionally, the security requirements are directly related to the functionality of the system-to-be. Use case-oriented security requirements methods such as misuse cases (Sindre and Opdahl, 2005 [354]) or abuse cases (McDermott and Fox, 1999 [264]) are one possible step before executing our method.

13.5. Conclusion

In this chapter, we introduced a methodology for Problem-based Security Requirements Elicitation (PresSuRE). PresSuRE is a method for identifying security needs during the requirements analysis of software systems using a problem frame model. It allows the requirements engineer to break down high level security goals ("The overall system needs to be secure") to actual security requirements related to the functionality of the system-to-be. In summary, the PresSuRE method has the following advantages:

- It introduces attacker asset access graphs and the graphs they are based on, which
  - visualize the access flows given by the functional requirements,
  - show entry points for attackers and subsequently threatened assets,
  - directly relate functional requirements and threats they are exposed to, and
  - can be generated fully automatically.
- And it is a re-usable requirements security analysis method which
  - relies on functional requirements only, as all security aspects are added while conducting the method in a structured way,
  - ensures that crucial domain knowledge is elicited,
  - enforces and supports the collaboration with system stakeholders and security experts,
  - is applicable to different domains, and
  - is tool supported to ease the elicitation, modeling, and analysis necessary for the method.

We validated our method and tool with two real-life cases in the fields of smart grid and voting systems. The results show the suitability of our method to detect initial security requirements.

Chapter 14

Patterns for Legal Compliance Requirements Elicitation

The goal of the step "Compliance Requirements Elicitation" is to bring together legal experts and software and system developers to identify relevant laws for a system-to-be. This step depends on the previously identified and described requirements for the system-to-be. The output of this step are laws and sections of these laws that are relevant for the given development problem and the requirements derived from these laws. Figure 14.1 shows the works this chapter is based on (**bold**) and the related work for this chapter (*italics*).

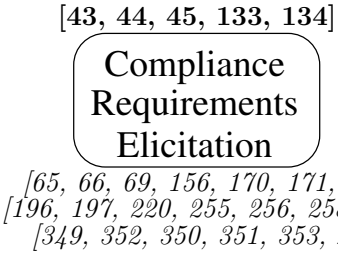


Figure 14.1.: *Compliance Requirements Elicitation*

14.1. Overview

In general, every legislator demands from everyone who lives in or is active within the jurisdiction of the legislator to comply with the laws the legislator enacts. Hence, software engineers have to assure that the system to be developed is compliant to all relevant laws of the jurisdictions the system-to-be will touch. Therefore, they need to know the legal requirements for the system to be developed. Based on this knowledge the engineers can decide whether and how to address compliance.

In the following chapters (Chapter 14, Chapter 15[1], Chapter 16[2], and Chapter 17[3]), we present the results of our research on legal compliance requirements elicitation. This chapter starts with Section 14.2, in which we motivate the conducted research and highlight the importance of compliance for IT systems in general and for requirements engineering in particular. Next, we review the state of the art in the field of legal compliance requirements engineering in Section 14.3. In Section 14.4[4], we introduce our Law Pattern and Law Identification Pattern, which are the basis for identifying relevant laws and deriving legal requirements. These patterns capture the knowl-

[1] Page 267
[2] Page 281
[3] Page 307
[4] Page 252

edge of legal experts. To gain this knowledge, we reviewed and analyzed different documents of the legal domain in order to understand how legal experts structure and analyze laws with a specific case in mind, which they want to judge. Section 14.4[5] is based on Beckers, Faßbender, Küster, and Schmidt (2012 [43]), and Beckers, Faßbender, and Schmidt (2012 [44])[6]. In the following Section 14.5[7], we introduce the general process which makes use of the Law Pattern and the Law Identification Pattern. This section gives the general overview of all steps and the inputs and outputs of these steps, while the subsequent chapters explain the steps in detail. The process presented in Section 14.5[8] was introduced in Beckers, Faßbender, and Schmidt (2012 [44])[9] for the first time and used in Beckers, Côté, Faßbender, Heisel, and Hofbauer (2013 [45]), Faßbender and Heisel (2013 [133]), and Faßbender and Heisel (2014 [134]). We conclude this chapter in Section 14.6[10].

The preparatory steps for conducting the process presented in Section 14.5[11] are detailed in Chapter 15[12], while the steps executed for every system-to-be are explained in Chapter 16[13]. Finally, we present and discuss the validation of our solution for legal compliance requirements engineering presented in this thesis in Chapter 17[14].

14.2. Motivation

In today's world many products and services are highly dependent on software and information systems. With the growing importance of IT systems over the past decades, legislators worldwide decided to regulate and enforce laws regarding IT systems. The permanently evolving technology inevitably leads to ever increasing legal requirements and regulations with severe penalties for non-compliance. Breaux and Baumer (2011 [68]) report their observation that every bigger data breach incident that is widely noticed by the public leads to a corresponding reaction by the US legislators. Afterward, there is a significant increase in law cases which are heard considering the changed regulations. The same development can be noticed, for example, for Germany. A couple of big data breach incidents in 2008[15] led to a law amendment of the federal data protection act (Bundestag der Bundesrepublik Deutschland (Lower House of German Parliament), 2009 [88]) in 2009. Many emerging information-driven businesses deploy information services without adequately considering legal issues. But compliance is critical for such systems as they are governed by regulations and law especially given that non-compliance can result in both financial and criminal penalties (Breaux and Baumer, 2011 [68]).

One prominent example is the German electronic voting system ban from 2009 (Federal Constitutional Court of Germany, 2009 [140]). The German Federal Constitutional Court decided

[5] Page 252

[6] The initial Law (Identification) Patterns were a contribution of the author, but revised in many discussions. The result of this discussion were fed back by the author to the patterns.

[7] Page 262

[8] Page 262

[9] The process is a contribution of the author.

[10] Page 264

[11] Page 262

[12] Page 267

[13] Page 281

[14] Page 307

[15] Just to name to the most prominent (Bundesbeauftragter für den Datenschutz und die Informationsfreiheit, 2008 [83]):

Deutsche Telekom 17 Million customer data records were stolen.

Deutsche Bahn The company spied on 774 of their own managers and their families.

Banks of Berlin The banks of Berlin shared customer information without any compliance to the federal data protection act.

on March 3, 2009 that an electronic voting system used for the 2005 general elections was not compliant to the relevant laws. Hence, the usage was unconstitutional and therefore the system must not be used for future elections. The Second Senate decided that the use of electronic voting machines requires that the essential steps of the voting and of the determination of the result can be examined by the citizen reliably and without any specialist knowledge of the subject. This requirement results from the principle of the public nature of elections (§38 in conjunction with §20.1 and §20.2 of the Basic Law (Grundgesetz – GG (Bundestag der Bundesrepublik Deutschland (Lower House of German Parliament), 2010 [89])). Beside the major concerns about the transparency of the voting process itself, the senate also criticized some flaws in the voting system regarding the federal data protection act of Germany (Bundesdatenschutzgesetz - BDSG (Bundestag der Bundesrepublik Deutschland (Lower House of German Parliament), 2009 [88])). Although the voting computers were banned for future use, the results from the elections were sustained since no evidence proved errors in the results.

Another emblematic example is the case of the information broker "ChoicePoint" for which an identity theft of more than 163.000 consumers was reported in 2004. An assessment of the company's products indicated that the products were developed without proper security controls as mandated by the Fair Credit Reporting Act. Since ChoicePoint failed to comply with these regulations, it was fined $10 million in civil damages and $5 million for consumer redress. Further, the settlement requires ChoicePoint to implement new procedures to ensure that it provides consumer reports only to legitimate businesses for lawful purposes, to establish and maintain a comprehensive information security program, and to allow audits by an independent third-party security professional every other year until 2026 (Federal Trade Commission, 2006 [141]).

Violations similar to those two examples are considered to be due to the subjective interpretation (or even ignorance) of regulations by companies in the context of their information systems' landscape. Nevertheless, the economic and social implications of erroneous or tampered voting results or stolen personal data are staggering. Hence, legal regulations have to be considered during the design of a software system to ensure adherence to the law. Ultimately, the law has to obeyed for all software products, regardless of size, revenues or scale of the producer.

Beside the general legal context and the resulting financial and criminal penalties, there are also economic reasons for companies to take up and deal with the legal compliance topic. Recent studies show that both, compliance violations (Cavusoglu et al., 2004 [98]) and not sufficient preparations regarding upcoming regulations (Khansa et al., 2012 [213]), have a negative impact on the market share and value. Moreover, transparency regarding specific compliance topics, for example data protection, gain more and more importance as incentives to convince customers of and investors for an IT product and give a competitive edge against competitors (Zwick, 2006 [403]; Vehlow and Golkowsky, 2010 [381]; Unabhängiges Landeszentrum für Datenschutz Schleswig-Holstein, 2014 [375]). At the moment, this statement might only be valid for business-oriented IT products and for specific countries and domains, but there are also indications for a global trend. For example, the case of the whistle-blower Snowden, who published the spy activities of the NSA, had a significant impact on the world-wide sales and revenues of big US firms (Stern-Peltz and Armitage, 2013 [361]; SpiegelOnline, 2013 [360]). Thus, there is not only a need for compliant IT products to satisfy governments, which is an extrinsic motivation for developers of IT products, but also an economic dimension, which in in turn is an intrinsic motivation for developers of IT products.

But the identification and analysis of relevant laws is considered to be difficult because it is a cross-disciplinary task in laws as well as in software and systems engineering (Biagioli, Mariani, and Tiscornia, 1987 [57]). Otto and Antón (2007 [302]) conclude in their survey about research on laws in requirements engineering that there is a need for techniques to identify relevant laws based on requirements, analyze them, and to derive requirements from them.

For our research we focused on the German law, as Germany is relevant for the use cases

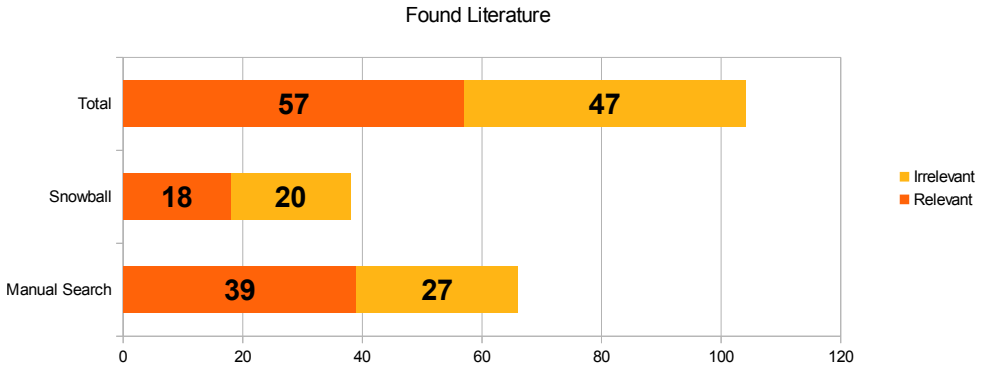


Figure 14.2.: *Found Literature*

which initially led to investigating the compliance topic. Moreover, the German law is one representative for the so called statute law (for an explanation see Section 14.4.1[16]), which is the common law system in Europe. Additionally, the scientific works covering requirements engineering and compliance mainly consider the US law, which is a case law (This claim is grounded in Section 14.3). Hence, here we discovered a gap in research regarding statute laws in general and the German law specifically.

14.3. RelatedWork

In the following we present and discuss the results of a problem & gap study in the field of legal compliance requirements engineering (Section 14.3.1). Afterward, we also present some important insights gained from reading the papers found within the problem & gap study and relate them to the overall results of the study (Section 14.3.2[17]). The actual discussion of the papers which are closely related to the solution proposed in this work is presented in Section 14.3.3[18].

14.3.1. Results of a Problem & Gap Study about Legal Compliance in Requirements Engineering

Some results from the analysis of the literature found for a problem & gap study about legal compliance in requirements engineering are presented in this section. The mapping study was conducted using a manual search in the publications of selected conferences and workshops spanning the years of 2009 until today[19]. The initially found literature was extended by snowballing without any limit. For more details see Chapter 4[20].

Some numbers regarding the search are shown in Figure 14.2. After scanning the titles and abstracts of more than 8000 papers found while searching the selected venues, 66 papers were selected as potentially related to context elicitation in requirements engineering. Additionally, 38 were included after analyzing the bibliography of already included papers (snowballing). After reading the full papers, 39 papers of the manual search and 18 found while snowballing were selected as relevant papers. Hence, we found 57 relevant papers in total[21].

Figure 14.3 shows the distribution over time of papers dealing with the topic of legal compliance requirements engineering. What we see from the graph is that the topic of legal compliance

[16] Page 252
[17] Page 246
[18] Page 249
[19] The last iteration was done in March 2015
[20] Page 67
[21] The list of relevant papers and the information collected for them is shown in the Appendix A.3.3

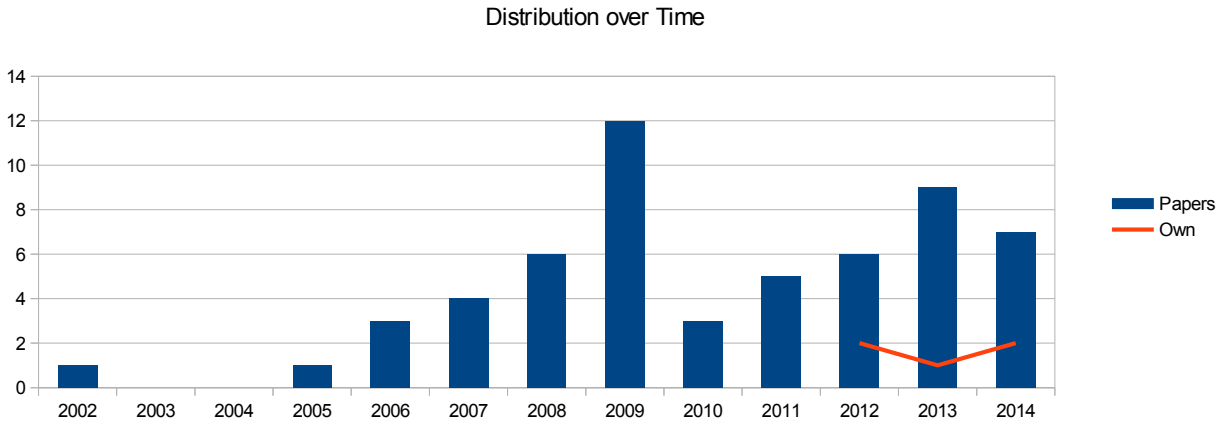


Figure 14.3.: *Distribution over Time of Found Literature*

is quite new in the field of requirements engineering. The first paper on this topic originates from the year 2002. Note that the topic of legal compliance for IT systems in general and software engineering in specific is not that new (see, for example, the work of Biagioli et al. (1987 [57])). The same applies for other compliance topics such as compliance with policies. Many papers on legal compliance in requirements engineering reference such works. These works were found during the literature review and are known to the author, but we decided to focus the problem & gap study on publications specific to the requirements engineering field. Hence, works from other fields do not show up in the results of our problem & gap study, even though some of them influenced the current state of the art in the field. From the found papers specifically focusing on legal compliance requirements analysis, we can state that the topic is quite new.

Starting in 2005, legal compliance became a hot topic with a peak in 2009. But even in the years after 2009 there is a significant amount of published work. One reason is the RELAW workshop which is held at the RE conference. This workshop is a major source for publications on the topic and has a very active community. Many solutions can be found as initial proposals at this workshop, and later on at other venues such as the RE in their final state. But the RELAW workshop is quite dominated by US-based researchers. Overall, we can state that the importance of the topic of legal compliance is nowadays well acknowledged in the requirements engineering community which manifests in many publications at different journals and conferences.

Table 14.1 shows the laws and activities covered by the proposed solutions in the found papers. Note that we only count those laws for which the solution was actually applied. Any assumption on the applicability on further laws were not taken into consideration. Of course, a solution can cover more than one law and activity. Whenever we summed up some rows or a column, we removed double counts. For example, in the second last row and the second column, the summed up value is not 24 but 17 as solutions proposed some papers cover different laws which leads to a double count, and in consequence to a count of 24 whereas the count without such double counts is 17. Whenever a number is given in brackets, this indicates papers which were published by us. Note that there are also some papers which cover an activity, but not a particular law. For example, only four of the five presented frameworks were actually applied to at least one law. Overall, Table 14.1 allows a quick overview which activities and laws are covered by the state of the art in the field of legal compliance requirements engineering.

We found nine different activities covered by the publications we found during the literature search. Papers about *modeling a law* investigate the matter of modeling a law using a requirements engineering notation or a new developed notation. For the majority of the papers the motivation for conducting the modeling is to derive legal compliance requirements from these models. When *modeling requirements*, the majority of papers aim at modeling requirements

		Modeling Law	Modeling Requirements	Elicitation	Identification	Decision Making	Interaction Detection	Framework	Analysis Law	Analysis Requirements	Total Papers (without double counts)	% of All Papers (57)
	Brazilian Information Access Law	1									1	1.75%
Canada	Canada Personal Information Protection and Electronic Documents Act	3	3	1			1	2	1	3	7	12.28%
Canada	Canada Commodity Futures Trading Commission Regulation	1							1		1	1.75%
Canada	Canada Aviation Law	1							1		1	1.75%
	Canada Total (without double counts)										*9*	15.79%
EU	EU Eudralex	1									1	1.75%
EU	EU 95/46/EC (Data Protection)	1					1			1	1	1.75%
EU	EU Civil Aviation Regulation		1							1	1	1.75%
	EU Total (without double counts)										*3*	5.26%
Germany	German Data Protection Law	5 (5)	5 (5)	5 (5)	5 (5)			1		2 (2)	6 (5)	10.53%
Germany	German Passport Law	2 (2)	2 (2)	2 (2)	2 (2)						2 (2)	3.51%
Germany	German Federal State Election Law	2 (2)	2 (2)	2 (2)	2 (2)						2 (2)	3.51%
Germany	German Electronic Signature Law	2 (2)	2 (2)	2 (2)	2 (2)						2 (2)	3.51%
	Germany Total (without double counts)										*6 (5)*	10.53%
Italy	Italian Law on EHR					1					1	1.75%
Italy	Italian Privacy law		1	1		1					3	5.26%
Italy	Italian Stanca Act			1							1	1.75%
	Italy Total (without double counts)										*4*	7.02%
	Japan Trade Secret Law		1								1	1.75%
	Luxembourg Tax Law	1			1						1	1.75%
US	US Data Breach Notification Law									2	2	3.51%
US	US Medical Record Retention Law									1	1	1.75%
US	US Sarbanes-Oxley Act	1		1						1	2	3.51%
US	US Health Insurance Portability and Accountability Act	5	4	6	3	1	5	1	2	7	21	36.84%
US	US Gramm-Leach-Bliley Act						2				3	5.26%
US	US Accessibility Standards										1	1.75%
	US Total (without double counts)										*26*	45.61%
	Total Papers (without double counts)	17 (5)	17 (5)	13 (5)	9 (5)	2	5	5	4	15 (2)		
	% of All Papers (57)	29.82%	29.82%	22.81%	15.79%	3.51%	8.77%	8.77%	7.02%	26.32%		

Table 14.1.: *The Laws and Activities Covered by the Found Papers*

derived from laws, and extending existing requirements models this way with legal compliance requirements. The activity *elicitation* is actually about the concrete way of deriving and eliciting legal compliance requirements, while the activity *identification* covers the identification of relevant laws for a system-to-be. *Decision making* is about selecting requirements or choosing solutions for making a system-to-be compliant. As also requirements drawn from one law or different laws can contain interactions with each other, the activity of *interaction detection* is concerned with identifying these interactions. A *framework* describes a complete legal compliance requirements engineering life cycle without giving details for each step within the life cycle. Last, there are two kinds of analysis in this area. When conducting an *analysis of a law*, this law is investigated in detail for a certain property, for example, the cross reference to other laws. Such an analysis is concerned only with the law itself. In contrast, an *analysis of requirements* is concerned with investigating derived legal requirements and the requirements of the system-to-be. An example is the analysis of a goal tree in which we try to determine if all

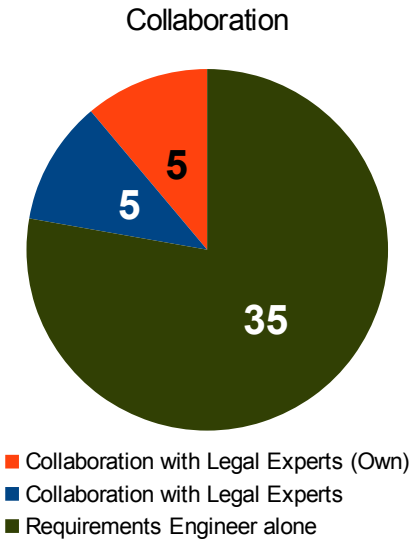


Figure 14.4.: *Papers Explicitly Proposing The Collaboration with Legal Expert*

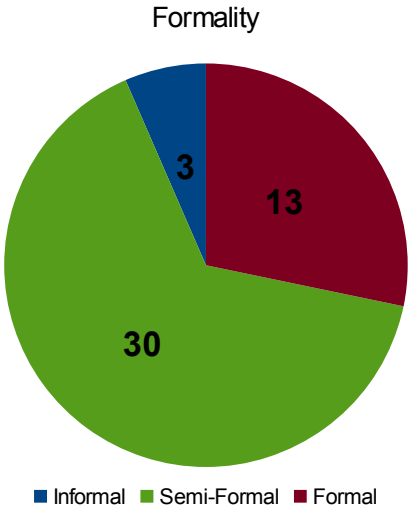


Figure 14.5.: *Grade of Formality*

goals are actually fulfilled.

Investigating the activities covered, we discover that some activities are considered and treated by many publications while others seem to be neglected. Modeling laws and requirements, eliciting legal requirements, and analyzing the requirements are the dominant activities covered in literature. Interaction detection, framework, and law analysis are covered to some extent by the existing literature. Neglected are the activities of law identification (leaving aside our publications) and decision making. We will discuss the implications of this observation in Section 14.3.2.

Taking a closer look at the covered laws, we see a noticeable imbalance regarding the countries and laws covered. 46 percent of all solutions were applied to US laws and in particular over a third of the proposed solutions cover the US Health Insurance Portability and Accountability Act (HIPAA). The very next country regarding the coverage by solutions is Canada with 16 percent. These two countries share the same law system (case law), they also share the predominant type of statute (prescriptive statutes), and they share the idea of sectoral, industry related organization of laws. In consequence, almost 3/4 of all papers proposing a solution were validated for laws coming from countries using this particular combination. The implications of this observation is a topic we will discuss in Section 14.3.2, too.

A further observation regarding the laws covered is that taking our papers aside, only one publication covering German law was existent. But there are some more publications covering EU law or law from countries within the EU. In total, the EU is as well covered as Canada, but we have to keep in mind that laws and law systems within the EU can differ from country to country, whereas Canada is a homogeneous jurisdiction.

Figure 14.4 highlights another interesting fact. Even though legal compliance RE is a cross disciplinary topic, only a few solutions proposed in papers directly require or enable the collaboration with legal experts. The majority of 35 papers relies solely on activities carried out by requirements engineers. Only 5 other papers beside ours highlight the necessity or require to collaborate with legal experts. We will also discuss the implications of this observation in Section 14.3.2.

We also collected the grade of formality of the solution proposed in a paper. Here, we distinguish between informal solutions which only require to use notations such as pure text, semi-formal solutions which use graphical notations with a semi-formal character for, for example,

modeling purpose such as the UML, and formal solutions requiring the requirements engineer to apply a formal notation such as first order logic. The result of this analysis is shown in Figure 14.5. It shows that most solutions in the field are semi-formal (30), but there is also a good share of formal solutions (13). Note that especially within the semi-formal solutions, the grade of formality differs. On the one hand some notations used provide many elements with very precise semantics, while on the other hand some notations are rather high-level, allowing a high degree of informality. We will discuss the consequences of (in)formality in the next section.

14.3.2. Important Insights Drawn From the Problem & Gap Study

While analyzing the literature of the field, some problems were constantly reoccurring in different papers. Hence, they define the problem space one has to consider when designing a solution for legal compliance engineering.

First of all, laws are formulated in natural language. Hence, they are prone to the same defects as natural language requirements such as ambiguity, incompleteness and so forth (Hassan and Logrippo, 2009 [171]). Especially the ambiguity within laws is seen as one of the mayor problems in the field of legal compliance RE (Hassan and Logrippo, 2009 [171]; Maxwell, Anton, Swire, Riaz, and McCraw, 2012 [260]; Massey et al., 2010 [256]; Darimont and Lemoine, 2006 [111]; Massey, Rutledge, Anton, and Swire, 2014 [253]; Breaux, Antón, Boucher, and Dorfman, 2008 [70]; Kiyavitskaya, Krausová, and Zannone, 2008 [219]). Hence, it is challenging to derive and formalize requirements from laws, which is necessary for many requirements engineering methods (Hassan and Logrippo, 2009 [171]). Despite the ambiguity, laws are written using a generic wording to cover many different cases which makes it hard to derive concrete requirements (Siena et al., 2012 [353]). Additionally, laws use a domain specific wording (specific to the legal domain as well as the regulated domain) which has to be be known by the ones analyzing a law (Maxwell et al., 2012 [260]; Massey et al., 2010 [256]; Zeni, 2015 [401]; Darimont and Lemoine, 2006 [111]; Alva, 2014 [15]; Breaux et al., 2008 [70]). Also in general, the wording, methods, and concepts used within the legal domain and the requirements engineering domain are different (Siena et al., 2009 [350]; Siena et al., 2008 [349]; Kiyavitskaya et al., 2008 [219]).

Beside the problems faced when analyzing one statement of a law, the great magnitude of relations between statements within one law, and between different laws is identified by several recent papers as a problem (Gordon and Breaux, 2013 [160]; Maxwell et al., 2012 [260]; Maxwell, Antón, and Earp, 2013 [261]; Massey et al., 2010 [256]; Breaux and Gordon, 2013 [69]). And even when one has identified all legal requirements, the measurement of the degree to which a system is compliant remains a problem (Tawhid, Braun, Cartwright, Alhaj, Mussbacher, Shamsaei, Amyot, Behnam, and Richards, 2012 [367]; Rashidi-Tabrizi, Mussbacher, and Amyot, 2013 [318]).

In consequence of all of these problems, the ability of requirements engineers to extract legal requirements has shown to be poor (Breaux, 2009 [67]; Massey et al., 2014 [253]). The same applies to the ability of requirements engineers to decide whether a requirement is already compliant, which also has shown to be poor (Massey, Smith, Otto, and Antón, 2011 [257]). These inabilities are due to the inadequate knowledge about the legal domain (Maxwell and Antón, 2009 [258]), but also due to the poor tool support provided for fulfilling the task. Whenever there is tool support, it helps to improve the outcome of legal compliance RE (Massey et al., 2010 [256]; Tawhid et al., 2012 [367]; Rashidi-Tabrizi et al., 2013 [318]). But still, most methods available are conducted purely manually and there is a lack of tools (Sapkota, Aldea, Younas, Duce, and Bañares-Alcántara, 2012 [337]). Hence, requirements engineers are nowadays not able to conduct legal compliance RE sufficiently.

Another challenge is the dynamism of laws and law interpretation, which make a continuous adaption of legal requirements necessary (Boella, Humphreys, Muthuri, Rossi, and van der Torre, 2014 [62]; Gordon and Breaux, 2013 [161]; Ghanavati, Amyot, and Peyton, 2009 [157];

Siena et al., 2009 [352]; Massey et al., 2010 [256]; Ishikawa et al., 2009 [196]; Kiyavitskaya et al., 2008 [219]). Moreover, laws and legal systems of different countries differ, but an international company has to be compliant to all of them (Gordon and Breaux, 2011 [159]; Zeni, 2015 [401]; Alva, 2014 [15]; Kiyavitskaya et al., 2008 [219]). Especially, the differences between prescriptive and outcome-based laws are identified as a problem (Rifaut and Ghanavati, 2012 [320]). While prescriptive laws allow only a small number of solutions, the compliance to an outcome-based law can be achieved using various solutions (Ingolfo, Siena, Jureta, Susi, Perini, and Mylopoulos, 2013 [194]).

Another problem stated is that small companies cannot afford to apply complex solutions for identifying legal requirements (Gordon and Breaux, 2011 [159]; Massey et al., 2010 [256]).

The question now is, if those problems are adequately treated by existing solutions or if there are any problems neglected by the legal compliance RE community. One interesting paper to answer this question was written by Boella et al. (2014 [62]), in which legal experts discuss the current state of research in the field of legal compliance RE. In the following we will discuss their statements, compare them with our own results, and relate them to statements found in other papers.

The first central topic discussed by Boella et al. (2014 [62]) is the topic of ambiguity.

> "Laws portray exemplary cases and do not have the ambition of covering all possible future situations, which though they may be unforseen, still need to be covered by the law. Lawyers have to interpret the law to adapt to new situations, in accordance with the goals the legislation was developed to achieve. In this context, ambiguity is not always regarded as a problem in legal practice."(Boella et al., 2014 [62])

Hence, from a legal point of view ambiguity is often intended to be part of statements within a law. In consequence, this ambiguity cannot be removed without introducing errors.

> "Formal models often fail to resolve ambiguities in natural language representations and simply result in unambiguously wrong specifications. (Kamsties, Berry, Paech, Kamsties, Berry, and Paech, 2001 [208])" (Boella et al., 2014 [62])

> "Formal approaches, even if they could allow consistency checking and automated reasoning between different requirements, fall short of being acceptable to legal practitioners, because they poorly reflect the dynamics of legal reasoning in practice, which uses all the richness of natural language."[22](Boella et al., 2014 [62])

Boella et al. (2014 [62]) draw the conclusion that any kind of formality has to be treated with care as it might lead to legal compliance requirements which might not be accepted in a legal case. But here the requirements engineers have a completely different perspective as they are used to see ambiguity as a defect, which has to be resolved. For example, Massey et al. (2010 [256]) state:

> "Identifying, classifying, and resolving the ambiguity found in the requirements is one of the most challenging and rewarding aspects of our methodology."

This leads to many solutions which apply formal methods to deal with legal compliance as we have already seen in Section 14.3.1 (Figure 14.5). A good share of the solutions is completely formal, and most of the solutions are semi-formal. From the point of Boella et al. (2014 [62]) also semi-formal solutions are questionable as long as they aim at removing ambiguity. To some extent, also researchers in the field of legal compliance RE started to acknowledge this view recently:

[22]Note that in this context already most of the semi-formal notations used are regarded as too formal by Boella et al. (2014 [62]).

> "Many approaches to resolving ambiguity in software engineering rely on disambiguation or removal of ambiguity. These may simply not be an option for software engineers addressing ambiguity in a legal text."(Massey et al., 2014 [253])

Hence, it is a challenge for researchers to find a balance between some kind of formality which allows compliance checking, specifications and so forth, but to avoid the tendency to see ambiguity as a problem which has to be removed by the requirements engineer.

Another central criticism of Boella et al. (2014 [62]) is formulated in the following quote:

> "More worrying, perhaps, is that in most papers the role of law is addressed without the involvement of legal practitioners, resulting in naive views of the Law. This makes the resulting systems difficult for legal practitioners to accept and use in evidence in case of dispute about the compliance of software."

This quote is astonishing on the first sight, because in the beginning of this section we discussed the matter of the inability of requirements engineers conducting legal analysis and that researchers in the field are aware of it, which should reflect in the solutions they propose. But still, Boella et al. (2014 [62]) did not find evidence that these insights have an impact on proposed solutions. Our problem & gap study revealed the same, as only a small number of solutions explicitly demand the collaboration with legal experts (see Figure 14.4). In general, Boella et al. (2014 [62]) state that the RE community falls short in including the practices of other fields regarding legal compliance, which is inevitable for such a cross cutting topic.

RE researchers seem to be aware of the importance of context within legal compliance analysis:

> "Standard RE techniques ... provide insufficient context to adequately capture and prioritize legal requirements." (Massey, Otto, and Antón, 2008 [254])

They are also aware of the necessity to interpret laws (Ishikawa et al., 2009 [196]). Still, Boella et al. (2014 [62]) observed that the solutions proposed tend to analyze statements within a law one by one. But analyzing a law statement by statement is not enough. A statement always has to be seen in a relation to other statements from the same law, other laws, cases, legal practice, the context given by the system-to-be at hand, and so forth (Boella et al., 2014 [62]). It is not sufficient to extract those relations alone, but also to interpret them in the context of the system-to-be. Here, Boella et al. (2014 [62]) state:

> "While the RE community acknowledges that legal interpretation is a fundamental part of legal practice, this issue remains confined to future work sections, even in the most recent literature (Breaux and Gordon (2013 [69]))."

As a final conclusion, (Boella et al., 2014 [62]) state:

> "We need a harmonizing methodology grounded in the reality of a juristic conceptualization of the law that a) promotes a sufficient level of acceptability among legal practitioners to facilitate relevant applications that would transfer academic research to legal industry and b) promotes dialogue to enable IT professionals to appreciate the complexities of the judicial process."(Boella et al., 2014 [62])

This is seconded by the experience of Darimont and Lemoine (2006 [111]), and Soltana, Fourneret, Adedjouma, Sabetzadeh, and Briand (2014 [357]) working on legal compliance in the industry.

Beside discussing the observations of Boella et al. (2014 [62]) in the context of our problem & gap study, we want to highlight some observations we additionally made. First of all, it is often stated that traceability between requirements and legal texts (Massey et al., 2008 [254]) is important. But still, most solutions do not establish and maintain such relations. Most solutions we found in our study were deriving requirements directly without considering the

existing requirements, or they were changing existing requirements models without adding the legal reasoning behind. This is a serious issue as it makes it hard to argue in favor of the actions taken to make a system-to-be compliant. But being compliant not only means to take action and ensure compliance on the system side but also to be able to prove what has been done to achieve compliance and to reason about its sufficiency in a legal case. Hence, requirements must be traceable to the legal statements they originate from and the other way round to be able to take a defendable position at court.

A second observation made was the imbalance in the treated legal systems (see Table 14.1). As we have already discussed in Chapter 2[23], legal systems and how laws are written can differ to a large extent. In consequence, a solution working for one country might not work for other countries. From our experience, this is the case for the solutions proposed for the US when they are applied for the German law. Some of them rely on (legal) language specific rules which obviously cannot work in a different language, but also many of them use concepts which are not commonly used in German laws. For example, most solutions for the US and Canada are applied to prescriptive laws by the researchers. As a result, those solutions require some descriptions of the *how* to achieve compliance. But all German laws we analyzed were outcome-based. They only describe what has to be achieved to be compliant. Hence, the solutions for the US and Canada are not applicable for Germany. In consequence, it is a serious issue that we observed a focus on certain countries and even more that we observed a focus on specific laws.

To summarize the results of the problem & gap study, we present some central aspects a legal compliance requirements engineering solution should cover:

Collaboration A key factor of a sufficient legal compliance requirements engineering solution is the involvement of legal experts. Only legal experts are able to decide when and how a requirement has to change to be able to take a stand in a legal case.

Ambiguity A solution has to take ambiguity within legal statements into account, but should not try to remove the ambiguity by using, for example, formal methods. Instead, the interpretation of the statement in the context of the system-to-be has to be done by legal experts involved when applying the legal compliance requirements engineering solution.

Bridge the Gap A solution has to take into account the practices of both domains, the requirements engineering domain as well as the legal domain. As those domains are very far apart from each other, a solution has to close the gap between those two domains to enable collaboration.

Wording The wording of the legal domain and the requirements engineering domain differ to a large extent. Moreover, the wording within a law is specific to this law. The same applies to the wording used within the requirements for a specific system-to-be. A solution should enable the applying experts to extract those wordings and relate them to each other.

Relations Relations between statements in a law, between laws, and so forth are important. Hence, they have to be extracted and explicitly considered when conducting legal compliance requirements engineering.

Traceability There needs to be a link between the requirements of the system-to-be and the laws to which the system-to-be shall be compliant. This is of particular importance in a legal case to be able to prove that all reasonable actions were taken to assure compliance.

Dynamics of laws A solution has to be able to cope with changes in laws. A change in a law should trigger an iteration of the solution. But this iteration should need a significantly lower effort to be taken than for initially applying the solution.

[23] Page 27

Tool support Many tasks within legal compliance requirements engineering are repetitive and / or require an overview about complex situations. Hence, a solution benefits a lot from tool support which supports the applying experts. Purely manual solutions raise the odds of mistakes and lower the acceptance of the solution.

Applicability A solution should clearly reflect the specifics of the legal systems and the laws it is meant for. This applicability has also to be clearly stated to enable potential users to judge whether the solution works in their case or not.

14.3.3. Legal Compliance Requirements Engineering

In the following, we will not discuss all of the literature found while conducting the problem & gap study, but those which cover some of the activities or laws which also our solution covers.

Hassan and Logrippo (2013 [172]) (Preliminary results and ideas of their solution were presented in Hassan and Logrippo (2008 [170]), and Hassan and Logrippo (2009 [171])) present a solution for modeling legal requirements as well as the requirements of the system-to-be in Alloy (a first order logic). For the modeled requirements of the system-to-be can it then be checked to which extent they already cover the legal requirements. Hassan and Logrippo (2013 [172]) state that one central step within their solution is to refine both, the legal requirements as well as the requirements of the system to be, to a level which is suitable to be modeled in a formal way. Hassan and Logrippo (2013 [172]) solution was tested using laws from Canada and the US. For this solution the problem of formalizing laws clearly applies. Additionally, the solution is applied by requirements engineers alone. Like our solution, the solution of Hassan and Logrippo (2013 [172]) allows to intertwine the requirements of the system-to-be with legal requirements. But unlike our solution, the solution of Hassan and Logrippo (2013 [172]) does not identify applicable laws, but requires the applying requirements engineer to already know the relevant laws.

Siena et al. (2008 [349]), and Siena et al. (2009 [350]) describe the differences between legal concepts and requirements. They model the regulations using an ontology. The ontology is based on the Hohfeld taxonomy (Hohfeld, 1917 [186]), which describes the means and relations between the different means of legal texts in a very generic way. Thus, Hohfeld does not structure a certain law at all, but aims at the different meanings of laws. Hence, the resulting process defined by Siena et al. (2008 [349]) to align legal concepts to requirements and the given concepts are quite high-level and cannot be directly applied to a scenario. In further works (Siena et al., 2009 [352]; Siena et al., 2009 [351]), the authors try to bridge the gap between the requirements engineering process and compliance using a goal-oriented method. In these works they propose to derive goals from regulations and apply those goals to the actors within a requirements engineering scenario. They tested their approach using laws from Italy and the US. As they only derive goals from the laws, the solution is reported to work for both legal systems. In contrast to our method, they do not identify relevant laws, do not intertwine compliance regulations with already elicited requirements, and do not integrate legal experts in their method.

Ghanavati et al. (2007 [156]) present a framework for achieving compliance with privacy laws. In particular, they focus on how to connect goal models, business processes and laws by adding new goals, actors, and tasks, which originate from a law, to an existing goal and process model. For each of the newly added elements a traceability link to the law it originates from is added. The solution was validated using a Canadian law. This solution requires one to know the applicable laws beforehand. All the steps of the framework are conducted by requirements engineers alone. Moreover, Ghanavati et al. (2007 [156]) report that especially for the processes it often happens that removing ambiguity is necessary to get a precise process.

Siena et al. (2012 [353]) present a language to model laws. The starting point of their modeling approach are certain, so-called *situations* which capture a scenario which is described by the

law and which makes the law applicable. The situations themselves are described using natural language. Along with the situations, one has to extract duties and rights from the law document. These three elements are then connected using a formal language (Nomos 2), which allows one to describe different relations between situations, situations and duties / rights, and between duties and rights. The modeled law is then used to check whether a system-to-be has to comply to this law. This is achieved by selecting those situations which can also take place while using the system. The selected situations are then used to do a formal reasoning to determine which duties and rights are of relevance. There are some differences to our solution. First of all, the solution of Siena et al. (2012 [353]) is conducted by requirements engineers alone. The situations (facts of the case in our solution) are only captured in natural language, while we model them more precisely capturing the most important elements. In consequence, our solution builds up a dictionary of important legal terms, which can be connected to the wording of the system-to-be, while the solution of Siena et al. (2012 [353]) does not provide such a mapping. Moreover, Siena et al. (2012 [353]) do not describe how to derive the situations from the requirements of a system-to-be, while our solution provides guidance for this part. Nevertheless, the solution of Siena et al. (2012 [353]) might be combined with our solution as their language is more expressive regarding the rights and duties and their relation to the situation. If this expressiveness is also suitable for our approach can be questioned as Nomos 2 was used to model the HIPAA law from the US which is a detailed prescriptive law. In case, this formalization does not require precise descriptions of the "how to comply" or to remove ambiguity, adding this language to our solution might improve the matching and reasoning.

Maxwell and Antón (2009 [258]), and Maxwell and Antón (2009 [259]) developed a method to check existing software requirements for regulatory compliance, i.e., to discover violations and missing requirements. They model a law using formal production rules which are codified using Prolog. To check whether a law applies or not requires the requirements engineer to formulate for each requirement of the system-to-be a query which describes the requirement in a specific, formal way. Those queries are then used to obtain the applicable laws. Maxwell and Antón (2009 [258]) state that understanding the legal terms and mapping them to the actual terms used to express the requirements is crucial to successfully write queries, but they do not provide any guidance on this part. This solution was tested using the US law HIPAA and is meant to be applied by a requirements engineer alone. Additionally, the drawbacks of formal approaches regarding laws apply to this approach.

Based on an initial work (Kiyavitskaya et al., 2008 [220]), Zeni (2015 [401]) presents a tool called GaiusT which can be used to extract rights and obligations from legal texts in an (semi)-automated way. The tool takes a legal text as input, analyzes it, and highlights the legal text in a way that it is easy to identify obligations and rights for a user. To work properly, the tool needs several inputs which define important words used within a legal system which, for example, indicate a right, or specific grammar constructs. They validated the tool using the US HIPAA and the Italian Stanca Act. This tool might also be of use for our solution to ease the work of modeling a law.

The work of Ishikawa et al. (2009 [196]) deals with the problem of interpreting laws to refine requirements that are established directly or indirectly by these laws. Therefore, law interpretations are modeled in a meta model, and stepwise refinement relationships are used to derive concrete goals from high-level goals given by the laws. As an example the paper focuses on Japanese trade secret laws. As our work focuses on identifying and structuring laws, this work might complement ours by eliciting concrete requirements from laws by helping to interpret them by the presented meta model.

Breaux and Gordon (2013 [69]) propose a semi-formal specification language to express legal requirements. The purpose of this language is to formulate requirements which are directly derived from a law. Afterward, the requirements of one law as well as requirements of different

laws can be analyzed for, for example, coherence. The purpose of the language is to enable legal experts to write regulations in a way that they can be easily used by requirements engineers afterward. From our point of view, such an approach might be usable for a prescriptive law, such as the HIPAA, but for laws which do not describe the how but only the outcome, such as the German law, it is not applicable. Nevertheless, Breaux and Gordon (2013 [69]) language also contains constructs which are similar to the one we use to describe the facts of a case. Hence, it might be worthwhile to investigate, based on the proposed language, how our approach can be used to cover prescriptive laws.

Islam et al. (2010 [197]) propose a framework for eliciting and managing security and privacy requirements from laws. This work proposes some generic steps and highlights the information needed to conduct this step. Additionally, the authors outline the relation between the initial and the information derived when conducting the steps. The solution of Islam et al. (2010 [197]) was tested using the German BDSG. Our solution fits into the proposed framework, as it details important steps of the solution of Islam et al. (2010 [197]).

Breaux et al. (2006 [66]), and Breaux and Antón (2008 [65]) present a framework for analyzing the structure of laws using a natural language pattern. This pattern helps to translate laws into a more structured restricted natural language and then into a first-order logic. The modeled law can then be checked for interacting requirements defined by the law and so forth. In contrast to our work, the authors do not support identifying laws based on functional requirements of a system-to-be. The framework also suffers from the drawbacks of formal logic analysis of laws, as discussed in Section 14.3.2.

Massey et al. (2009 [255]) devise a method to find legal implications in software requirements and to prioritize the requirements according to a score computed based on the found implications. The authors find legal implications in requirements by a discussion between requirements engineers and legal domain experts. The score is the sum of legal implications in a requirement. The proposed solution of Massey et al. (2009 [255]) and our solution can be used complementary. On the one hand, Massey et al. (2009 [255]) do not provide means for easing the discussion between the requirements engineers and legal experts like our solution does. But on the other hand, analyzing the requirements, which are important for being compliant, regarding their impact on the level of compliance is not covered by our solution. In another paper, Massey et al. (2010 [256]) give a methodology for improving existing system requirements that must comply with relevant laws. Again, this method is complementary to ours, because it does not identify the requirements, which are of importance to be compliant, but, in contrast to our solution, discusses how to actually align the identified requirements to be compliant.

14.4. Patterns for Describing Laws and Law Identification

Pattern-based approaches capture the knowledge of domain experts. In this way, the knowledge is made explicit and can be re-used for recurring problems. And the problem of law identification is recurring for each system-to-be while the solution used by legal experts remains the same. Hence, we propose a pattern-based method for identifying and analyzing laws. The patterns allow the identification of relevant laws for a system-to-be based on its requirements.

The German law is a *statute law* in the tradition of the Roman jurisdiction. Statute laws are specified by the legislator and written down in legal documents. Hence, every judgment of a court is based exclusively on the analysis of the legal documents relevant for the judged case (Schwacke, 2003 [344], p. 41). We analyzed how judges and lawyers are supposed to analyze a law, based upon legal literature research. These insights lead to a basic structure of laws and the contained sections, which we used to create a Law Pattern. We describe the results of this analysis in the following.

14.4.1. General Structure of Laws

First of all, a law is a textual document. This law document is structured into *sections*. Each section defines a legal aspect of the law and contains several statements. These statements are *dictates of justice*, so-called *legal rules* (Larenz, 1983 [230], p. 240). There are different types of dictates of justice. Complete and self-contained dictates of justice are one type. This type is the fundamental building block of every law (Larenz, 1983 [230], p. 241).

A full dictate of justice is divided into the *facts of the case*, the setting which is regulated, and the *legal consequence*, the resulting implications of the setting (Beaucamp and Treder, 2011 [35], p. 7). Furthermore, a dictate of justice has also an *addressee(s)*. The reason is that every complete dictate of justice is an *imperative*, or can be transformed into an imperative (Larenz, 1983 [230], pp. 243-44), and an imperative has to be directed towards an addressee(s) (Schwacke, 2003 [344], pp. 3-4). Most complete dictates of justice are refinements of other complete dictates of justice. Thus, these complete dictates of justice just add additional information or circumstances which detail the aspects of the general case. The most general complete dictate of justice is defined in the first sections of a law.

Besides the complete, self-contained dictates there are: (Larenz, 1983 [230], pp. 247-251)

- *definition dictates* that describe and refine terms and other basic elements.
- *restricting dictates*, which add exceptions to a complete dictate. A restricting dictate of justice does not define a law structure, but adds restrictions to the complete dictate of justice. Sometimes it also adds additional legal consequences.
- *referring dictates*, which reference one or more dictates. The referenced dictates contain (parts of) the facts of the case or the legal consequences. A referring dictate always adds further information to a complete dictate of justice. This might be information about competing regulations, which have to be discussed, complementing regulations, which have to be obeyed at the same time as the connected dictate of justice, substitutional dictates of justice which can be obeyed instead of the connected dictate of justice, or just a further dictate of justice which contains additional law structure elements or legal consequences.
- *fiction dictates*, which equate different facts of the case. The equated facts have to be treated as similar for judging a case even though they are different.

All of these dictates cannot be analyzed in isolation, as they have relations to other dictates (or even laws). The types of relation between these dictates are *refinement*, *addition*, and *constraint*. This implies that all of the resulting dictates and laws, and the relations between them, have to be considered when analyzing laws. A *regulation* is the set of rules applicable to a specific case for which the conflicts are resolved (Larenz, 1983 [230], p. 254).

Thus, relations between laws, sections and dictates of justice are of fundamental importance. They are arranged in a hierarchy, which is not always free of conflicts (Larenz, 1983 [230], p. 255). A special part of these relations is the terminology used within a jurisdiction. This terminology is organized as trees where the terms of the more general dictates of justices are refined by subsequent dictates of justice.

14.4.2. The Subsumption Method

Beside the structure of laws, it is also of relevance how legal experts, such as lawyers or judges, analyze these laws in the context of a specific law case. The legal experts use the *subsumption method* to analyze if a dictate of justice is applicable to a specific case. The general subsumption schema is shown in Table 14.2. To start the analysis, the legal experts formulate a *hypothesis*, which is relating the actual case and the full dictate of justice at hand. Afterward, they identify

Hypothesis	
Major Premise	
Subsumption	**Hypothesis**
	Major Premise
	Subsumption
	Conclusion
	...
Conclusion	

Table 14.2.: *Subsumption Schema (based on (Schwacke, 2003 [344], pp. 52-53; Larenz, 1983 [230], pp. 260-264; Zippelius, 2012 [402], 16 I-II; Engisch et al., 2005 [129], pp. 104f))*

a *major premise*, which must be checked to validate the hypothesis. The major premise is based on the facts of the case of the full dictate of justice at hand or related definitions. Next, they break down the initial hypothesis into sub-hypotheses in a top down manner until the basic elements of the facts of the case of the according major premise are reached. The basic elements of the facts of the case to be checked on the lowest level of the subsumption schema can have the four general kinds *Addressee*, *activity*, *target subject*, and *target person* (Definitions of these kinds are given in Table 14.6). Then, they match (*subsume*) the basic elements of the facts of the case with the elements of the actual case. They not only match the terms themselves but also check if the kind of the elements is matching. For example, a human is always classified the same, regardless if he / she is the addressee or the target person. But for checking the hypothesis, it is of relevance if the human is the one who has to be compliant or the one who has to be, for example, protected. Each *subsumption* leads to a *conclusion*. As long as every conclusion is positive (the hypothesis is valid), the conclusions are aggregated in a bottom up way until the initial hypothesis is reached. This way they check if the actual case covers the facts of the case of the full dictate of justice at hand. (Schwacke, 2003 [344], pp. 52-53; Larenz, 1983 [230], pp. 260-264; Zippelius, 2012 [402], 16 I-II; Engisch et al., 2005 [129], pp. 104f).

A company announces a competition for which the participants have to register with their full name and address. The CRM (customer relationship management) system is used for this purpose.

Table 14.3.: *Example Case*

(1) The purpose of this Act is to protect the individual against his/her right to privacy being impaired through the handling of his/her personal data. (2) This Act shall apply to the collection, processing and use of personal data by 1. ... 2. ... 3. private bodies in so far as they process or use data by means of data processing systems or collect data for such systems, process or use data in or from non - automated filing systems or collect data for such systems, except where the collection, processing or use of such data is effected solely for personal or family activities. ...

Table 14.4.: *BDSG Section 1 (Bundestag der Bundesrepublik Deutschland (Lower House of German Parliament), 2009 [88])*

Example:
Table 14.3 shows an example case for which we will analyze if the federal data protection act (BDSG) is of relevance. An excerpt of the first section of the BDSG is shown in Table 14.4. Table 14.5 shows parts of this analysis (see Appendix E.1[24] for the full subsumptiom result). We start with the *general hypothesis* "The company might have to ensure that specific rights and privacy needs of the participants, who are entitled to the registration data, are preserved.". The *general major premise* to validate this hypothesis is the BDSG Section 1. Next, the general hypothesis is broken down into two sub-hypotheses. *Hypothesis 1* is about the participants and their right on privacy. *Hypothesis 2* is about the company and the characteristics of collecting information using a CRM (customer relation management) system. To judge hypothesis one we have to break it down into further hypotheses. For example, *hypothesis 1.2* is about the personal nature of registration data. To analyze the hypothesis, we use the *BDSG Section 3*, which defines personal data, as *major premise 1.2*. To qualify any data as personal data it has to identify a participant (*hypothesis 1.2.1*) and it has to describe a circumstance of life of this participant (*hypothesis 1.2.2*). For both hypotheses, there is no major premise defined in the BDSG itself. Thus, we have to look for general, widely accepted definitions. In this case, we use *Black's dictionary* (Black and Garner, 1999 [58]) for finding definitions. Black's dictionary is one of the widely known and accepted legal dictionaries in the English speaking world. At this point, we have reached the end of the top-down break down of the general hypothesis. Now, we start the bottom-up aggregation of results, which includes the *interpretation* of a major premise under the light of the corresponding hypothesis. For example, we argue in *subsumption 1.2.1* that the full name of the participant can indeed identify that particular person. Hence, we conclude (*conclusion 1.2.1*) that the hypothesis is confirmed. We also confirm that the address contained in registration data describes personal circumstances (*conclusion 1.2.2*). Combining these two conclusions, we deduce that registration data is personal data (*conclusion 1.2*). We proceed in this manner till we reach the *general conclusion*. In our case, all hypotheses are confirmed. Thus, our general hypothesis is also confirmed and the company has to consider the federal data protection act.

If not all terms of the case to be judged can be mapped to facts of the case as defined by the dictate of justice at hand, the dictate of justice is not relevant for the case. However, a mapping between all terms and notions of the case and the basic elements is not sufficient to really prove the relevance of a dictate of justice for a case. The reason is that the specific case can contain elements that have no mappings to a term of the facts of the case of the dictate at hand. The subsumption solely considers a mapping from the dictate of justice to the terms of the specific case. The other direction is not considered. But such uncovered elements have the potential to prove that the law is not relevant for the specific case. The subsumption leaves this gap intentionally, because the mapping of specific cases to laws is based upon human interpretation in the end. Indeed, the subsumption relies in many steps on human interpretation and only identifies candidates which might be relevant laws and sections. To prove the actual relevance an interpretation heavy legal revision needs to be done. In such a revision, legal experts also consider already judged cases, the relation between different, possibly competing laws, and so forth. This crucial need for interpretation shows that dealing with laws always includes the human factors. Thus, only parts of a legal analysis can be subject of computer-aided support.

[24] Page 485

Hypothesis	The company might have to ensure that specific rights and privacy needs of the participants, who are entitled to the registration data, are preserved.			
Major Premise	**(BDSG Section 1)** The purpose of this Act is to protect the individual against his/her right to privacy being impaired through the handling of his/her personal data. … This Act shall apply to the collection, processing and use of personal data By … private bodies in so far as they process or use data by means of data processing systems or collect data for such systems …			
Subsumption	**Hypothesis 1**	The participants could be in danger of loosing privacy when the registration data is handled improperly.		
	Major Premise 1	**(BDSG Section 1)** The purpose of this Act is to protect the individual against his/her right to privacy being impaired through the handling of his/her personal data.		
	Subsumption 1	**Hypothesis 1.1**	A participant could be an individual.	
			…	
		Conclusion 1.1	*A participant is an individual.*	
		Hypothesis 1.2	Registration data, including full name and address, could be personal data	
		Major Premise 1.2	**(BDSG Section 3)** "Personal data" means any information concerning the personal or material circumstances of an identified or identifiable individual.	
		Subsumption 1.2	**Hypothesis 1.2.1**	The full name might identify an participant or might make him/her identifiable.
			Major Premise 1.2.1	**(Black's Law Dictionary)** *Identification:* Proof of identity; the proving that a person, subject, or article before the court is the very same that he or it is alleged, charged, or reputed to be; True identity is collected from a multitude of signs.
			Subsumption 1.2.1	The full name might not reveal the true identity in all cases. But it is one of the signs which can be used to derive the true identity.
			Conclusion 1.2.1	*The full name identifies an participant or makes him/her identifiable.*
			Hypothesis 1.2.2	The address might describe personal circumstances.
			Major Premise 1.2.2	**(Black's Law Dictionary)** *Personal Circumstances:* Information relating to the private aspects of a person's life.
			Subsumption 1.2.2	The information where somebody is living reveals private aspects of this person's life.
			Conclusion 1.2.2	*The address describes personal circumstances.*
			Registration data includes information about personal circumstances. And it identifies the related person or makes the person identifiable.	
		Conclusion 1.2	*Registration data, including full name and address, is personal data*	
		Hypothesis 1.3	Someone might be able to impair the privacy of the participant using the registration data.	
			…	
		Conclusion 1.3	*Someone can impair the privacy of the participant using the registration data.*	
		When the registration data leaks to unauthorized persons, the unauthorized person can tamper with the privacy of the participant.		
	Conclusion 1	*The participants is in danger of loosing privacy when the registration data is handled improperly.*		
	Hypothesis 2	The the company might be a private body processing registration data, which might be personal data, using its CRM system, which might be a data processing system.		
		…		
	Conclusion 2	*The the company is a private body processing registration data, which is personal data, using its CRM system, which is a data processing system.*		
	The company has to handle the registration data properly to protected the rights and privacy needs of the participants.			
Conclusion	*The company has to ensure that specific rights and privacy needs of the participants, who are entitled to the registration data, are preserved.*			

Table 14.5.: *Subsumption Example (Excerpt)*

14.4.3. The Structure of Complete Dictates of Justice

From the previously presented information we derived the structure of complete and self-contained dictates of justice and present the results in Tab. 14.6.

According to Section 14.4.1, a full dictate of justice is divided into the *facts of the case*, the setting which is regulated, and the *legal consequence*, the resulting implications of the setting. Furthermore, a dictate of justice has also an *addressee(s)*.

The facts of the case need to be further refined to be useful for a pattern later on. The legal method called *subsumption* contains a further refinement of the facts of the case. This refinement results in the *law structure elements activities*, *target subjects*, and *target persons* (Beaucamp and Treder, 2011 [35], pp. 23-31).

14.4.4. The Law Pattern

Based on the previously discussed structure of laws, we define a *law pattern* shown in Fig. 14.6. The pattern consists of three parts: the dark gray part represents the *Law Structure*, the light gray part depicts the *Classification* to consider the specialization of the elements contained in the *Law Structure* in related laws or sections, and the white part considers the *Context*.

The context part (white area in Fig. 14.6) of the law pattern contains the *Legislator(s)* defining the jurisdiction, and the *Domain(s)* clarifying for which domain the law was established.

Laws, sections, and dictates of justice are often interrelated. For instance, dictates might not contain all necessary elements to instantiate the *Law Structure* as they are a restricting,

Addressee(s)	has (have) to comply to the law.	
Facts of the case	**Activity(ies)**	describe(s) actions that an addressee has to follow or avoid to be compliant.
	Target subject(s)*	describes impersonal subjects that are objectives of the activity(ies). Subjects can be material, such as a product, or immaterial, such as information.
	Target person(s)*	are directly influenced by the activity(ies) of an addressee, or have a relation to the target subject(s).
Legal consequence	defines the consequence for an addressee, e.g. the punishment when violating the section.	

A * next to an element of the structure means that the element is optional.

Table 14.6.: *Structure of a Full Dictate of Justice*

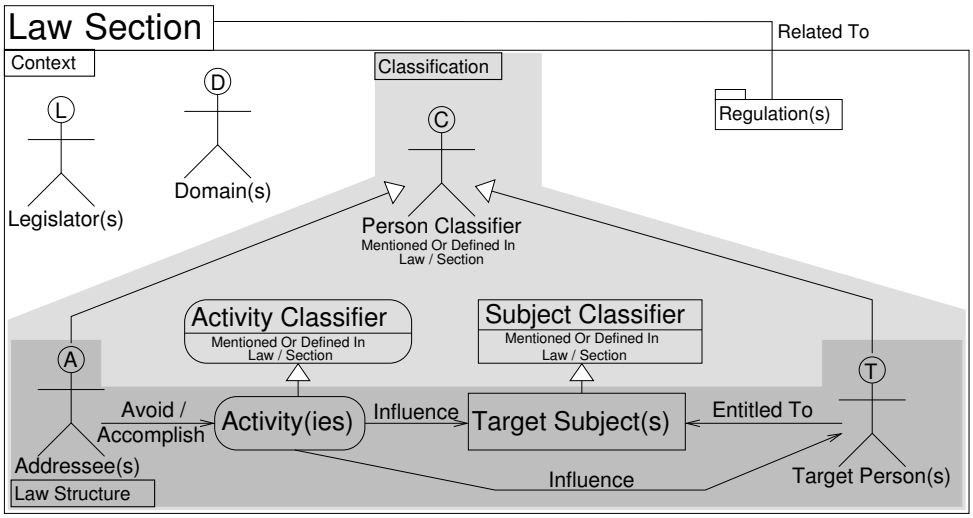


Figure 14.6.: *Law Pattern*

directing, or fiction dictate (see Section 14.4.1). Instead, the missing elements are defined or mentioned only in related sections or dictates of justice. Moreover, multiple definitions for the same term can be found in different laws. Hence, as it is necessary to know in which context and relation a law is used, we introduce *Regulation(s)*, which are *Related To* the section at hand. Those related regulations are also important for the identification of relevant laws, because matching one particular dictate of justice points out that also the related regulations might be of relevance.

The law structure part contains the elements as described by Tab. 14.6. Thus, the combination of *Addressee(s)*, *Activity(ies)*, *Target Subject(s)*, and *Target Person(s)* characterizes a dictate of justice and is the key information for a matching with requirements (The matching is described in detail in Section 16.3[25]).

For the matching, it is also important to know how the terms within a law are related. These relations are expressed in the classification part (light gray area in Fig. 14.6). In the classification part the *Person Classifier(s)* is (are) refined by the addressee(s) and target person(s), the *Activity Classifier(s)* is (are) refined by the activity(ies), and the *Subject Classifier(s)* is (are) refined by the target subject(s). While establishing these classifications for each dictate of justice of a law, a hierarchy of law structure elements used in this particular law is constructed. The process of building these hierarchies is described in Section 14.4.4.

We organize the already mentioned hierarchies (see Section 14.4.1) of person classifier, activity

[25]Page 298

classifier, and subject classifier using a tree structure. Figure 14.7 shows the generic structure of such hierarchies. A law structure element can refine another law structure element of the same type. For example, the *person classifier "refinement 1"* refines the *person classifier "1st root for law"*. While this example shows a refinement within one law, it is also possible that a law structure element refines a law structure element of another law. Such a case is the *activity classifier "1st root for law"* which refines the *activity classifier "refinement z"*. "refinement z" is part of another law. A law structure element is a root law structure element for a law, if it does not refine any other element of the law. It is possible to have several root law structure elements of the same type for a law. For example, *"1st root for law"* and *"2nd root for law"* are both *subject classifiers* and a root law structure element for the same law. It is also possible that a law structure element refines more than one other law structure element. For example, the *subject classifier "refinement 1"* refines *"1st root for law"* and *"2nd root for law"*. Examples for full law structure element hierarchies for a law are shown in the appendix E.3[26]. The law structure element hierarchies, which follow the generic structure as visualized in Fig. 14.7, are built while modeling a law, as described in Section 14.4, and used for matching, as described in Section 16.3[27].

Regulation(s), legislator(s), and domain(s) can be also ordered in hierarchies, similar to the classifiers. For instance, Germany is part of the EU and consists of several states.

We now describe one example instance for our Law Pattern using BDSG Section 1 as an example. The text of this particular section is shown in Tab. 14.7. The process of instantiating the Law Pattern is explained in Section 15.1[28]. The resulting instance is shown in Fig. 14.8.

In the *Context Part* (depicted as white area in Fig. 14.8), the *Legislator(s)* and *Domain(s)* are instantiated as *Germany* and *General Public*. The related *Regulation* is, for example, instantiated as *GG Section 2* in the *Context* part. GG Section 2 is a fundamental dictate of justice, which defines the right of informational self-determination. This relation is a cross-law reference. The other related regulations like, for example, BDSG Section 3a, are law-internal references.

[26] Page 490
[27] Page 298
[28] Page 267

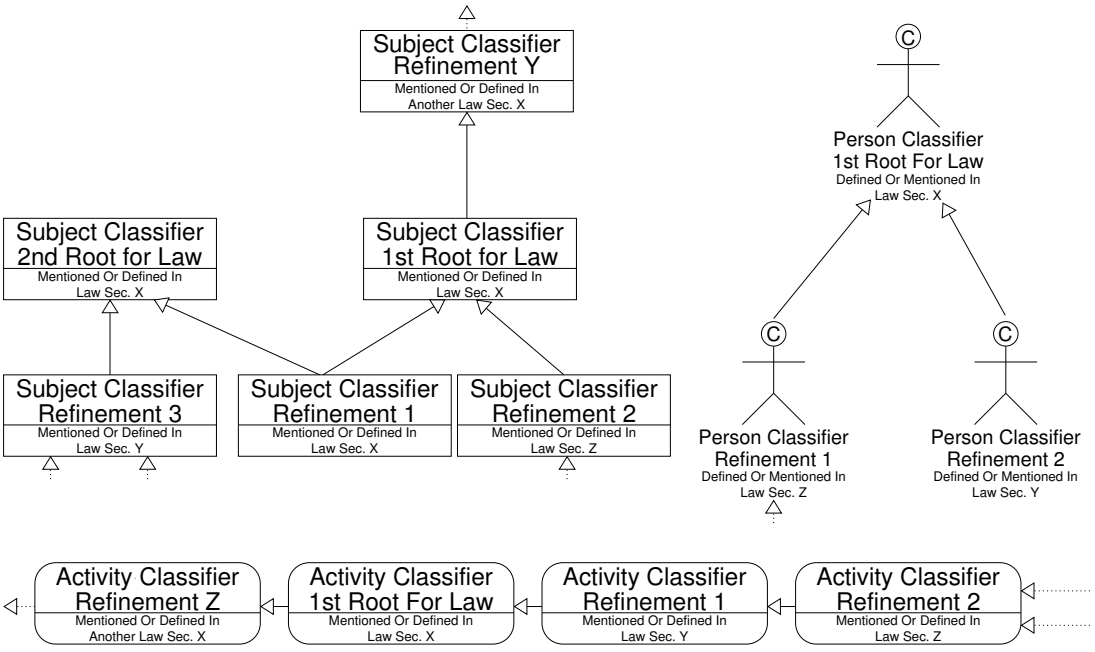


Figure 14.7.: *General Structure of Law Structure Element Hierarchies*

(1) The purpose of this Act is to protect the *individual* (**Target Person**) against his/her right to privacy being impaired through the handling of his/her *personal data* (**Target Subject**).
(2) This Act shall apply to the *collection, processing and use* (**Activities**) of personal data by

1. *public bodies of the Federation* (**Addressee**),
2. *public bodies of the Länder* (**Addressee**) in so far as data protection is not governed by Land legislation and in so far as they
   - a) execute federal law or,
   - b) act as bodies of the judicature and are not dealing with administrative matters,
3. *private bodies* (**Addressee**) in so far as they process or use data by means of data processing systems or collect data for such systems, process or use data in or from non - automated filing systems or collect data for such systems, except where the collection, processing or use of such data is effected solely for personal or family activities.

...

Table 14.7.: *BDSG Section 1 (Bundestag der Bundesrepublik Deutschland (Lower House of German Parliament), 2009 [88])*

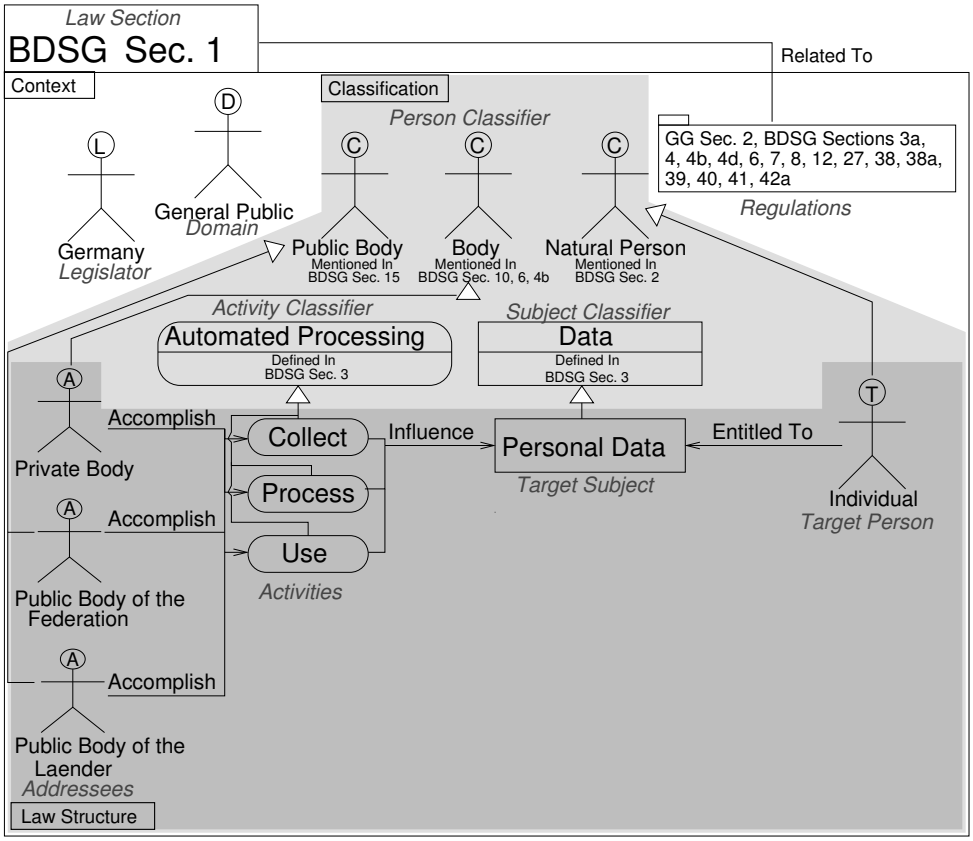


Figure 14.8.: *Law Pattern Instance for BDSG Section 1*

The *Law Structure* (depicted as dark gray area in Fig. 14.8) contains the central elements of the particular dictate of justice at hand. BDSG Section 1 defines *Private and Public Bodies of the Federation and Länder* as *Addressees*. The BDSG Section 1 is of relevance whenever one of the addressees accomplishes a *Collect*, *Process*, or *Use Activity*. Additionally, those activities have to influence *Personal Data* as *Target Subject*. An *Individual* as *Target Person* has to be entitled to the personal data.

For the *Classification* part (light gray area in Fig. 14.8), the private bodies are classified using the *Person Classifier Body*, which is mentioned in BDSG 4b, 6, 10. The public bodies of the

Federation and Länder are classified as *public body*, which is mentioned in BDSG Section 15. The activities collect, process, and use refine the *ActivityClassifier Automated Processing*, which is defined in BDSG Section 3. The *Subject Classifier Data*, which is defined in BDSG Section 3, is refined by personal data. The individual is not a public or private body, but a *Natural Person*.

14.4.5. The Law Identification Pattern

Identifying relevant laws based on functional requirements is difficult, because functional requirements are usually too imprecise, they contain important information only implicitly, and use a different wording than in laws. For example, a functional requirement such as "The customer can buy content." might lead to different laws when searching for "customer" only, and most of them might not be relevant in this case (e.g. laws dealing with retail markets). Additionally, laws dealing with online shopping, copy-right, or Internet communication are not taken in consideration since the functional requirement does not contain adequate keywords (e.g. content is not specific enough). Moreover, it is difficult to discover that the information needed for the payment process implicitly contains personal data, which has to be protected for privacy reasons. Formulating the requirement in a more comprehensive way, e.g. "The media market customer wants to buy content like music, films, and so forth, using a web interface providing payment data for the payment process.", does not solve these problems. Still, for some words the wording is too different from laws, such as web interface versus tele-media in Telemedia Act(Telemediengesetz (TMG)(Bundestag der Bundesrepublik Deutschland (Lower House of German Parliament), 2007 [87])), some words are too specific, such as payment data versus personal data in BDSG, and some words are misleading, such as customer who is not mentioned in BDSG or TMG but in the Store-closing Act (Ladenschlußgesetz (LadSchlG)(Bundestag der Bundesrepublik Deutschland (Lower House of German Parliament), 2006 [86])).

To bridge the gap between different wordings and to facilitate the discussion between requirements engineers and legal experts, we define a *law identification pattern* to support identifying relevant laws based on functional requirements of a system-to-be. We especially use the laws captured with the Law Pattern and the simultaneously established hierarchies of terms presented in the previous section (Section 14.4.4), and the knowledge collected in terms of requirements.

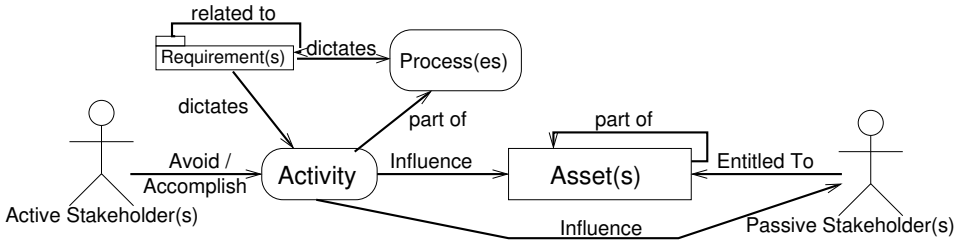


Figure 14.9.: *Relations between important elements which describe the system-to-be*

Important for the matching are the functional requirements, the activities the requirements contain, the assets which are important for the fulfillment of the requirements, and the relations between them and important domain knowledge. Figure 14.9 shows these relations. First of all, a *Requirement* can be related to other *Requirements* and dictates a certain behavior of the machine. A behavior can be a certain *Activity* or a whole *Process*. A *Process* consists of different *Activities*. An *Activity* is avoided or accomplished by an *Active Stakeholder* and influences an *Asset*. Additionally, an *Activity* influences a *Passive Stakeholder* in a direct way or indirectly through an *Asset* to which the *Passive Stakeholder* is entitled. In addition, *Assets* can be related to each other, e.g. one *Asset* is part of another *Asset*. All these relations have also to be

discovered, and the information has to be documented. In Section 16.1[29] we will explain how to discover and document this information. The documented information serves then as a basis for the law identification and is used for instantiating the Law Identification Pattern.

Figure 14.10 shows our Law Identification Pattern. The structure is similar to the Law Pattern (see Fig. 14.6) to allow a matching of instances of both patterns. In contrast to the legal vocabulary used in the *Law Structure* of our law pattern, the wording for the elements in the dark gray colored *Core Structure* of our Law Identification Pattern is based on terms known from requirements engineering. The *Active Stakeholder* corresponds to the *Addressee* of the Law Pattern, the *Activity* to *Activity*, *Asset(s)* to *Target Subject(s)*, and *Passive Stakeholder* corresponds to *Target Person*. The elements active stakeholder(s), activity, asset(s), and passive stakeholders form the core structure elements of a Law Identification Pattern.

The *Classification* part (light gray area) of the law identification pattern reflects the translation of terms specific for the system-to-be into legal terms. This is different from the classification part of the Law Pattern, as in case of the Law Pattern the classification part is used to build the law structure element hierarchies for the law at hand. The classification part of the Law Identification Pattern is used to connect the domain of the system-to-be with the legal domain. How to relate the core structure elements of a law identification pattern instance and the law structure elements of a Law Pattern instance is explained in Section 16.2[30]. Here, the subsumption step of relating the elements of the specific case with the law structure elements used for formulation dictates of justice takes place.

Our Law Identification Pattern takes into account that requirements are often interdependent (*Requirement(s)* in the *Context* part). Given a law relevant for one requirement, the same law might be relevant for the dependent requirements, too. Furthermore, the pattern helps to document similar dependencies for a given *Activity* using the *Related Process(es)* in the *Context* part. If one activity within a process is regulated by a law, it might be that the whole process or a subset of other activities within the process is regulated by the same law. To restrict the search for relevant laws later on, the context part of a Law Identification Pattern also contains the *Legislators* and *Domains* the system-to-be might be related to. For a legislator it means that the system-to-be or parts of it reside within the jurisdiction of the legislator. For a domain it means that the system-to-be is used in this domain, for example in a specific industry.

[29] Page 281
[30] Page 295

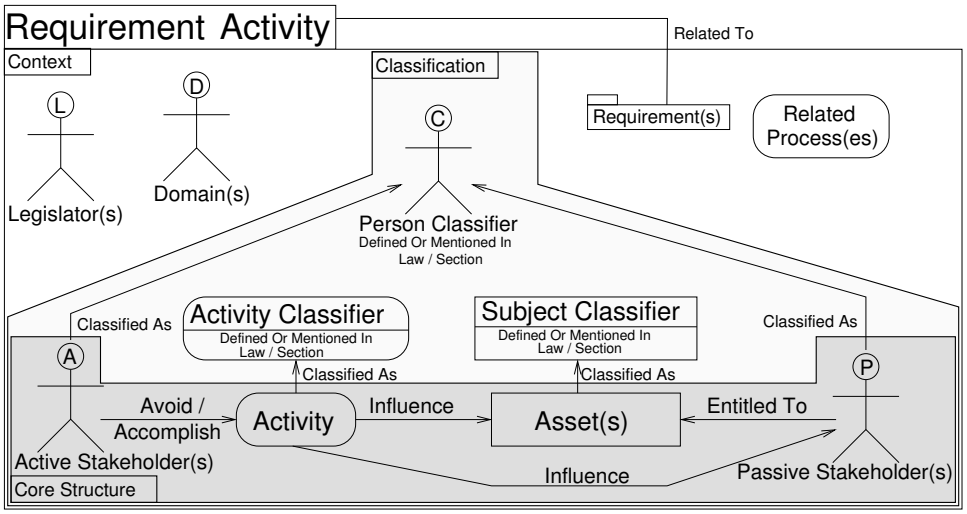


Figure 14.10.: *Law Identification Pattern*

(R23[34]) **Save Content Requests** "The media market saves content requests."

Table 14.8.: *Text for R23*

One example Law Identification Pattern instance for our media market example is given Fig. 14.11. The process of instantiating the Law Identification Pattern is explained in Section 16.1[31], and Section 16.2[32].

The core structure (dark gray area) depicts that the *Content Aggregator*, who is the active stakeholder, *saves the requests*, which are sent to his/her system. This activity influences the *Payment Data* as the payment data is part of each request. The passive stakeholder *Customer* is entitled to this asset. At this point we see that the original requirement (Table 14.8) only contains the activity. The rest of the information contained in the core structure is only given implicitly in the requirement. Thus, one challenge when using the Law Identification Pattern is to reveal the information that the media market operates on behalf of the content aggregator, and that each request contains payment data which is bound to a customer. In Section 16.1[33] we describe our solution for this challenge of discovering missing domain knowledge.

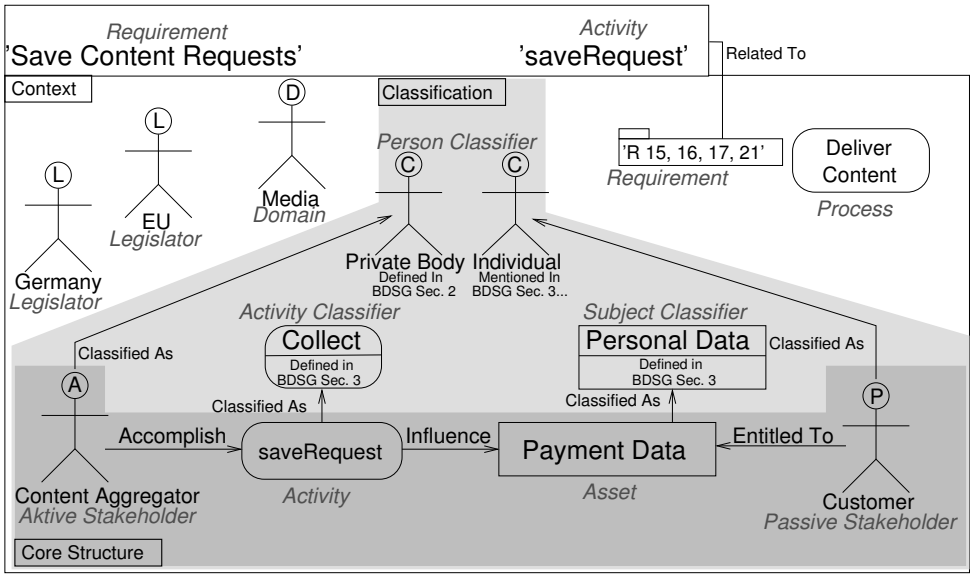


Figure 14.11.: *One Law Identification Pattern Instance for R23*

In the classification part (light gray area), we find the information that, in terms of the legal language, the content aggregator is a *Private Body*, the customer is an *Individual*, save request is *Collecting* information, and that the payment data is *Personal Data*. The process used for classifying core structure elements is described in detail in Section 16.2[35]. Note that for the classification of core structure elements the law structure elements of all modeled laws are used. Thus, it is not necessary to already know the relevant laws as long as one has access to a comprehensive database of law patterns. The matching of Law Identification Patterns and Law Patterns, which is described in Section 16.3[36], then reveals the candidates for relevant laws.

In the context (white area) it is depicted that the system-to-be will reside in *Germany* which

[31] Page 281
[32] Page 295
[33] Page 281
[35] Page 295
[36] Page 298

is part of the *EU*. Therefore, both define the jurisdictions the system-to-be has to be compliant to as legislators. The system to be will operate in the *Media* domain. The activity of saving request is part of the overall process of *Delivering Content*. Thus, some more requirements, namely R 15, 16, 17, and 21, which are also part of this process, are related to the requirement described by the Law Identification Pattern at hand.

14.5. Method

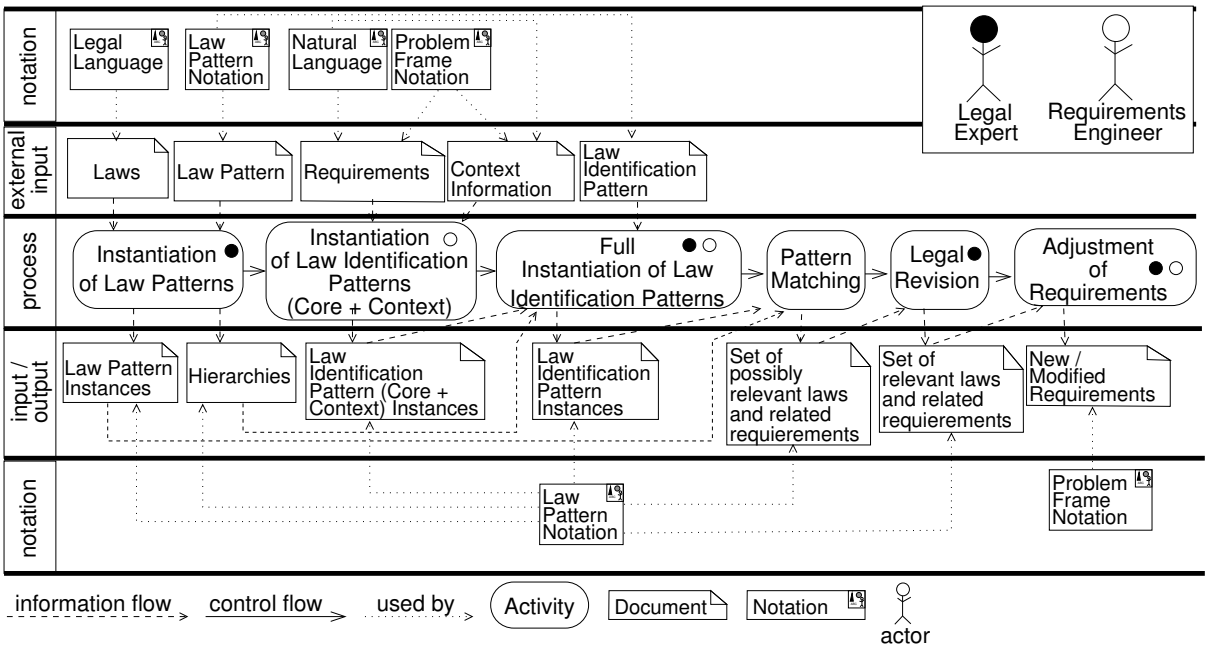


Figure 14.12.: *Law Identification Process*

The procedure for identifying relevant laws consists of six activities as depicted in Fig. 14.12. The first activity *instantiation of Law Patterns* (Section 15.1[37]) is about setting up a database of all laws which might be of relevance in general for IT systems. Therefore, *laws* have to be analyzed and stored in the structure of the *Law Pattern*. Thus, they are stored as *Law Pattern instances*. Additionally, the *hierarchies* of legal terms are also set up and stored. The laws are expressed in *legal language*, while the Law Pattern instances and hierarchies use the *Law Pattern notation* as introduced in Section 14.4.4 and Section 14.4.5. This activity is not needed if such an up-to-date database covering all possibly relevant laws already exists. The step of modeling the Law Pattern corresponds to the building of *major premises* used within the *subsumption* method (see Section 14.4.2[38]), with the difference that in case of our database we are preparing all premises, not only the needed ones for discussing a certain case.

Running Example:
Note that for the running example we only model and use the BDSG. In a normal usage scenario one would fill and maintain a broad range of laws. This would also be reflected in the hierarchies used as the law structure element hierarchies are built across all laws in the database. This is also true for the relations between laws and the contained sections.

[37]Page 267
[38]Page 253

The second activity *instantiation of Law Identification Pattern (core + context)* (Section 16.1[39]) uses information from the *requirements* of the system-to-be and their context to instantiate the core structure and the context of the *law identification pattern*. The requirements are given as *natural language texts* and the according problem diagrams using the *problem frame notation*. The resulting *Law Identification Pattern (core + context) instances* are expressed using the Law Pattern notation. This step corresponds to preparing the forming of *hypotheses* within the subsumption method by restructuring the actual case in way that the important facts are derived from the case.

Third (Section 16.2[40]), the relations between laws and requirements have to be established to prepare the identification of relevant laws for the given software for the activity *full instantiation of Law Identification Pattern*. Hence, a mapping between the terms of the requirements to legal terms is established. For the mapping the law identification pattern (core + context) instances and the hierarchies are used for the terms to be mapped and the Law Identification Pattern for the structure of the classification part. All inputs are modeled using the Law Pattern notation. The result of this activity are fully instantiated *Law Identification Pattern instances* given in the Law Pattern notation. This step corresponds to forming all possible *hypotheses* in the subsumption method by connecting core structure and law structure elements.

For the fourth activity *pattern matching* (Section 16.3[41]), the Law Pattern instances and Law Identification Pattern instances have to be matched. This results in a *set of possibly relevant laws and their relation to the requirements* of the system-to-be. The resulting laws are only possibly relevant, because our law pattern is based on the subsumption method. And the subsumption method overapproximates the relevance of laws for a legal case. Hence, our pattern matching also overapproximates the relations between requirements and laws. The possibly relevant laws and related requirements are expressed using our law pattern notation. The pattern matching corresponds to the actual "*subsumption* and *conclusion* deriving" in the subsumption method.

In consequence, the fifth activity *legal revision* (Section 16.4[42]) is about a detailed analysis of the matched laws and requirements. In this activity some laws and requirements are removed from the set of possibly relevant laws and related requirements which results in a *set of relevant laws and related requirements*. Such a step is also necessary when applying the subsumption method as described in Section 14.4.2[43].

Last, we have to perform an *adjustment of requirements* which is described in Section 16.5[44]. For this activity, we have to reflect the found laws in the original requirements. A law can be reflected either by a deletion of a requirement which is denied by the law, by adding a requirement which is demanded by a law, or by changing a requirement according to a law. In the end, we get a set of new or modified requirements expressed as problem diagrams in the problem frame notation.

For the process of law identification, law experts and requirements engineers have to work together for the necessary knowledge transfer. The first activity can be accomplished by legal experts alone, and for the second activity only requirements engineers are needed. But for the third activity both groups are needed to bridge the gap between the legal and the technical world. Afterward, the fourth activity can be performed automatically. The legal revision is then performed by legal experts alone, while the adjustment of requirements needs expertise from the legal and the requirements world.

[39] Page 281
[40] Page 295
[41] Page 298
[42] Page 302
[43] Page 253
[44] Page 302

14.6. Conclusion

In this chapter, we have motivated the general relevance of legal compliance for today's IT systems. This relevance is increasing as the legislators of different countries react to significant incidents in IT systems which have an impact on the citizens of a country, such as data breaches, by enacting new laws. In consequence, the number of law cases increases as well as the penalties for being non-compliant. But there are also other, economic, reasons such as market value which motivate companies to assure compliance of the products they sell or use. Based on this motivation, we took a look at the current state of the art in the field of legal compliance requirements engineering. Afterward, we have presented our patterns and their grounding in legal practice for tackling the problem of legal compliance in requirements engineering. Additionally, we outlined the general process to use these patterns. In particular, the contributions of this chapter are as follows:

- A problem & gap study reviewing the literature in the field of legal compliance requirements engineering. The study included
  - a mapping of existing literature to covered activities, laws, grade of formality, and if they propose a collaboration with legal experts,
  - a discussion of acknowledged problems which have to be tackled when doing legal compliance requirements engineering,
  - a critical investigation of the current state of the art and the shortcomings of existing solutions,
  - and a list of aspects a solution in the field of legal compliance requirements engineering should consider.
- We provided insights in the current practice of legal experts.
- We presented a pattern to model laws.
- We presented a pattern to restructure requirements in way that they are suitable for matching with laws.
- We presented a general process which enables requirement engineers and legal experts to collaborate and supports them in identifying relevant laws, and related requirements for a system-to-be.

Chapter 15

Preparing the Compliance Requirements Elicitation

The law identification process (see Figure 14.12) as presented in the previous chapter, contains steps which do not have to be conducted for each execution of the process, because they prepare inputs which can be reused. Hence, these steps are only conducted in case a new input has to be generated or an existing one has to be updated. This is the case for the instantiation of the Law Pattern. Once a law is modeled using the Law Pattern, it can be used for several executions of the law identification process. The step of instantiating the Law Pattern is explained in detail in Section 15.1. While the general idea of how to instantiate a Law Pattern was already presented in Beckers, Faßbender, Küster, and Schmidt (2012 [43]), the actual process with detailed advices considering all types of dictates of justice is novel work.

Another reusable input which has to be prepared once, are so called transformation cards which we propose for supporting the instantiation of core structures of Law Identification Pattern instances. These transformation cards can be used whenever the functional requirements of the system-to-be are modeled as problem diagrams. We introduce the idea of transformation cards in Section 15.2[1], which is based on Faßbender and Heisel (2013 [133]), and Faßbender and Heisel (2014 [134]).

15.1. Instantiation of Law Pattern

The process of instantiating Law Patterns, as depicted in Fig. 15.1, starts with a set of *laws* which have to be modeled. From this set, we *select a law* which is not modeled yet. For this *law*, we *identify* the according *legislator(s)* and the target *domain(s)*. The information about the legislator(s) is given in the preamble of the law text. Sometimes also the *domain(s)* are given in the preamble. In case the domain(s) are not defined in the preamble, they are defined in the first section containing a complete dictate at the latest. In some case no specific domain is mentioned. Then, the domain "general public" applies. The identified legislator(s) and domain(s) apply for all dictates of justice contained within a law. But it is possible that for some sections of the law the applying domain is refined. Hence, we assume for all sections the same domain(s) unless we find an explicitly mentioned refinement, which then applies only for this particular section.

Running Example:
In our case we select the Federal Data Protection Act (BDSG) (Bundestag der Bundesrepublik Deutschland (Lower House of German Parliament), 2009 [88]) as law to exemplify the process. As the the preamble states (see Table 15.1), the BDSG was enacted by the German Bundestag

[1] Page 275

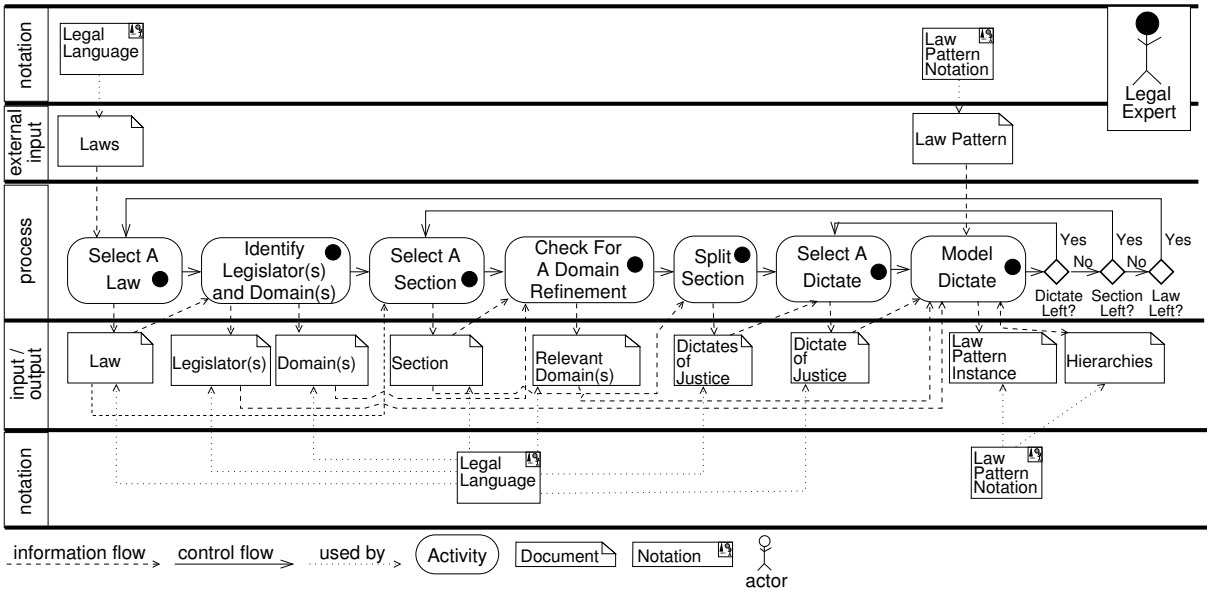


Figure 15.1.: *Law Pattern Instantiation Process*

with affirmation of the Bundesrat. Thus, the *legislator* is *Germany*. As no restriction for the target domain is given in the preamble and the first section, the *domain* of this law is the *General Public*.

Next, we *select a section* of the law at hand which was not modeled up to this point. Note that sections can contain subsections. For selecting the next section we do not distinguish between top level sections and subsections. We iterate over the sections in the order they occur in the document. For the selected *section*, we *check for a domain refinement*. In case we find such a refinement, we consider the refined domain(s) as *relevant domains* for the section at hand. In case we do not find a refinement, the domain(s) given by the laws are considered as relevant domains.

Running Example:
To exemplify our method for several types of dictates of justice as defined in Section 14.4.1[2] we select section 3 as it contains examples for definition dictates of justice, section 4b as it contains complete as well as referring dictates of justice, and section 4c as it contains constraining dictates of justice. When modeling a particular section we always assume that all preceding sections are already modeled. No section in the BDSG contains a domain refinement. Thus, the domain *General Public* applies as *relevant domain* for all sections.

[2]Page 252

This law was enacted . . . by the Bundestag (Lower House of German Parliament) with affirmation of the Bundesrat (upper house of the German parliament). . . .   . . .

Table 15.1.: *BDSG Preamble (Bundestag der Bundesrepublik Deutschland (Lower House of German Parliament), 2009 [88])*

...
(4) *"Processing"* (**Parent Activity**) means the storage, modification, transfer, blocking and erasure of personal data. In particular cases, irrespective of the procedures applied:

1. *"storage"* (**New Activity**) means the entry, recording or preservation of personal data on a storage medium so that they can be processed or used again,
2. *"modification"* (**New Activity**) means the alteration of the substance of stored personal data,
3. *"transfer"* (**New Activity**) means the disclosure to a third party of personal data stored or obtained by means of data processing either
   - a) through transmission of the data to the third party or (**New Activity**)
   - b) through the third party inspecting or retrieving data held ready for inspection or retrieval, (**New Activity**)
4. *"blocking"* (**New Activity**) means labelling stored personal data so as to restrict their further processing or use,
5. *"erasure"* (**New Activity**) means the deletion of stored personal data.

...

Table 15.2.: *Definition Dictate Of Justice contained in BDSG Section 3 (Bundestag der Bundesrepublik Deutschland (Lower House of German Parliament), 2009 [88])*

As a section can contain several *dictates of justice*, we *split the section* at hand into these dictates of justice. Then we *select* one *dictate of justice* which is not already modeled and proceed further. For the dictate of justice at hand we have to determine its type according to the types defined in Section 14.4.1[3]. The next activities to be taken vary depending on the type of the dictate of justice.

Running Example:
Section 3 contains fifteen definition dictates. Table E.3[4] and Table E.4[5] in Appendix E[6] show the full text of section 3 and the separated definition dictates of justice. Section 4b contains one complete, one referring and one restricting dictate and several legal consequences. Table E.2[7] in the appendix shows the full text of section 4b and the separated dictates of justice. Section 4c contains several restricting dictate and one legal consequences. Table E.5[8] in the appendix shows the full text of section 4c and the separated restricting dictates of justice.

15.1.1. Instantiation of a Definition Dictate of Justice

For a definition dictate of justice we *identify* the contained *law structure element(s)* as next step. Hence, we look for terms or formulations within the text of the dictate of justice which indicate *addresses, activities, target subjects, or target persons.* Then, we *add the new elements to the according element hierarchy.* This includes that we also add the refinement relations which might be implied by the dictate of justice at hand.

[3]Page 252
[4]Page 488
[5]Page 489
[6]Page 485
[7]Page 487
[8]Page 489

Running Example:
Table 15.2 shows the fourth definition dictate contained in section 3. It defines the *activities storage, modification, transfer, blocking, and erasure* which refine the parent activity *process*. The activity transfer is refined even further into the activities *transfer through transmission* and *transfer through inspection or retrieval*, which are only mentioned but not defined. Hence, we have to add these seven activities to the law structure element hierarchy for activities. Figure 15.2 shows the law structure element hierarchy for activities after adding the new activities. Activities previously defined are shown in white while added activities are highlighted in gray.

15.1.2. Instantiation of a Fiction Dictate of Justice

For modeling, a fiction dictate of justice can be treated like a definition dictate of justice as described in Section 15.1.1. Within the legal revision the definition and fiction dictates are treated differently. But this has no influence on the modeling.

15.1.3. Instantiation of a Complete Dictate of Justice

For a complete dictate of justice we *identify* the contained *law structure element(s)* as next step (see Fig. 15.3). Hence, we look for terms or formulations within the text of the dictate of justice which indicate *addresses, activities, target subjects, or target persons*. For all found law structure elements we check if they are already part of the according element hierarchies or if they are newly introduced elements. In the latter case, we *add the new elements to the according element hierarchy*. This includes that we also add the inheritance relations which might be implied by the dictate of justice at hand.

Running Example:
Table 15.3 shows the complete dictate of justice contained in section 4b. The dictate explicitly mentions the *activity transfer*, the *target subject personal data*, and the *target person bodies*. The target person and subject are influenced by the activity. The target person and subject were

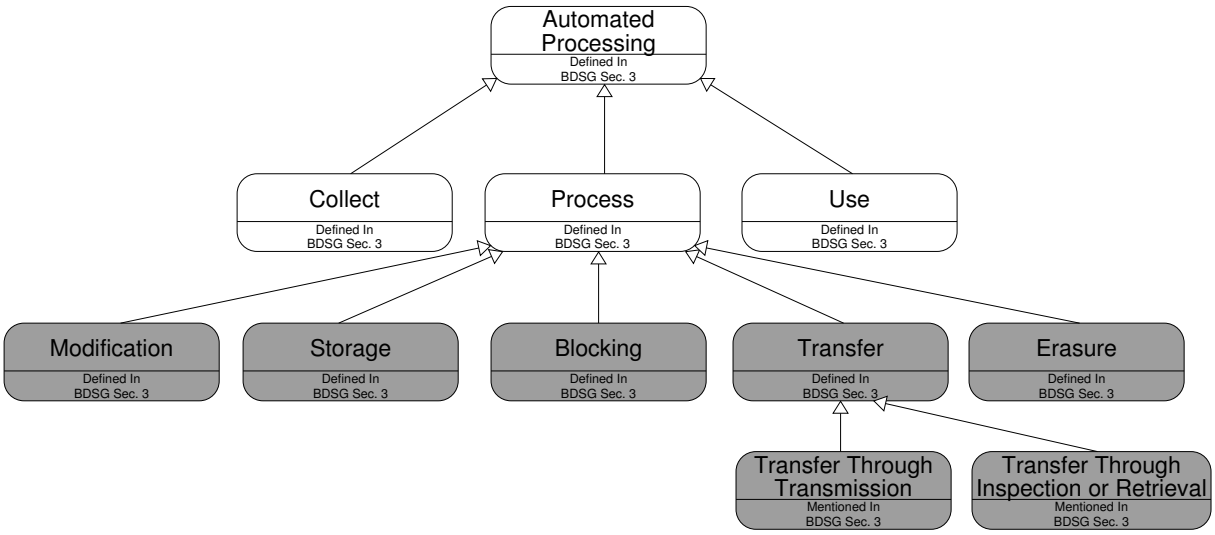


Figure 15.2.: *Updated law structure element hierarchy for activities after adding activities defined in BDSG Section 3*

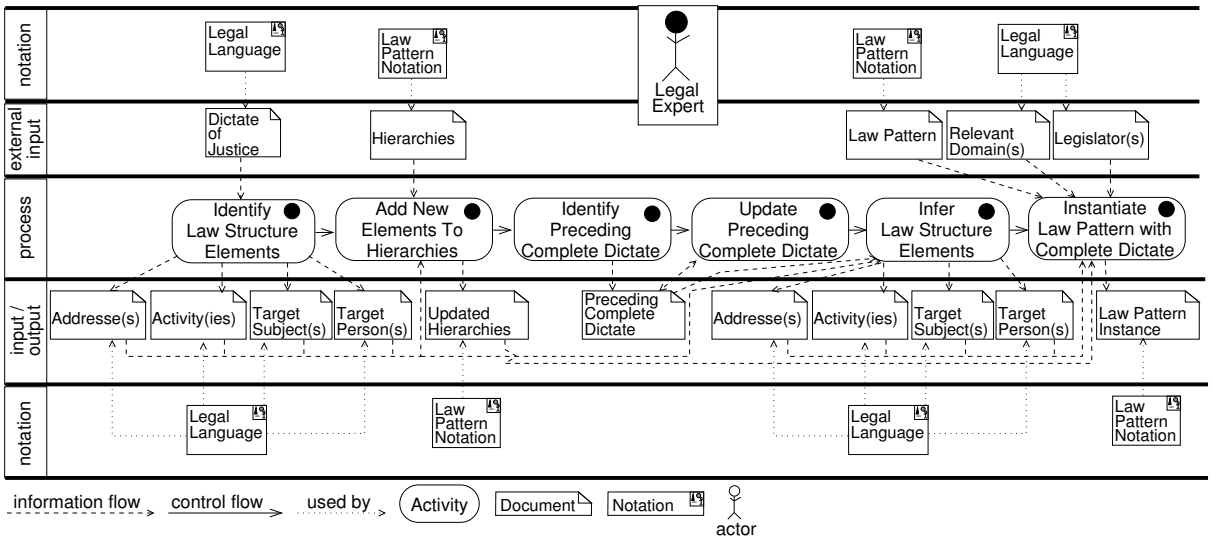


Figure 15.3.: *Model Dictate (Complete Dictate of Justice)*

(1) *The* transfer ***(Activity)*** *of* personal data ***(Target Subject)*** *to* bodies ***(Target Person)***
1. in other Member States of the European Union,
2. in other states parties to the Agreement on the European Economic Area or
3. institutions and bodies of the European Communities
(**Complete Dictate of Justice**) shall be *subject to Section 15 (1), Section 16 (1) and Sections 28 to 30a in accordance with the laws and agreements applicable to such transfer, in so far as transfer is effected in connection with activities which fall in part or in their entirety within the scope of the law of the European Communities.*(**Referring Dictate of Justice**) ...

Table 15.3.: *Complete, and Referring Dictates Of Justice, contained in BDSG Section 4b (Bundestag der Bundesrepublik Deutschland (Lower House of German Parliament), 2009 [88])*

already defined in preceding sections. Thus, the according hierarchies do not have to be updated (All full law structure element hierarchies as defined by the BDSG are included in Appendix E[9] and shown in Fig. E.1[10], Fig. E.2[11], and Fig. E.3[12]). The activity transfer mentioned is not the general transfer as already defined, but the special case of an abroad transfer which is not part of the law structure element hierarchy for activities yet. Hence, we have to add this activity. The abroad transfer is a refinement of the general transfer but also a refinement of the activities "transfer trough transmission" and "transfer through inspection or retrieval". Figure 15.4 shows the updated law structure element hierarchy for activities after adding the activity abroad transfer (highlighted in gray).

Then, we *identify the preceding complete dictate of justice.* Most complete dictates of justice are refinements of other complete dictates of justice. Thus, these complete dictate of justice just add additional information or circumstances which detail aspects of the more general case. The most general complete dictate of justice is defined in the first section of a law. The *preceding*

[9]Page 485
[10]Page 490
[11]Page 491
[12]Page 492

complete dictate of justice is explicitly mentioned in the dictate of justice at hand or it is the last complete dictate of justice defined in the preceding sections. The preceding dictate and the dictate of justice at hand are related regulations. Hence, we have to *update* the related regulations for the *preceding dictate of justice*.

Running Example:
The preceding complete dictate of justice is contained in section 1 of the federal data protection act. All sections in between do not contain any further complete dictates of justice. Hence, we update the Law Pattern for the complete dictate of justice contained in section 1 and add section 4b as related regulation as shown in Fig. 14.8[13] in Section 14.4.4[14].

In most cases, a complete dictate of justice which refines another complete dictate of justice only refines some elements. Thus, it might not contain all needed law structure elements explicitly. Hence, we have to *infer the missing elements* from the preceding complete dictate of justice. With inferring the missing elements we get the *complete law structure element(s)*.

Running Example:
Section 4b does not define the addressee. Hence, the addressees have to be inferred from section 1 as it is the preceding complete dictate of justice. As Fig. 14.8[15] shows, the *addressees* mentioned in section 1 are *"Public Body of the Federation", "Public Body of the Länder", and "Private Body"*. Thus, these addressees also apply for section 4b. We also infer an additional *target person* as section 1 mentions the *individual* as entitled to the personal data.

In the last step, we use these elements, together with the relevant regulations, domain(s), and legislator(s), to *instantiate the complete dictate of justice*. The classification part can be directly taken from the according element hierarchies.

[13]Page 259
[14]Page 256
[15]Page 259

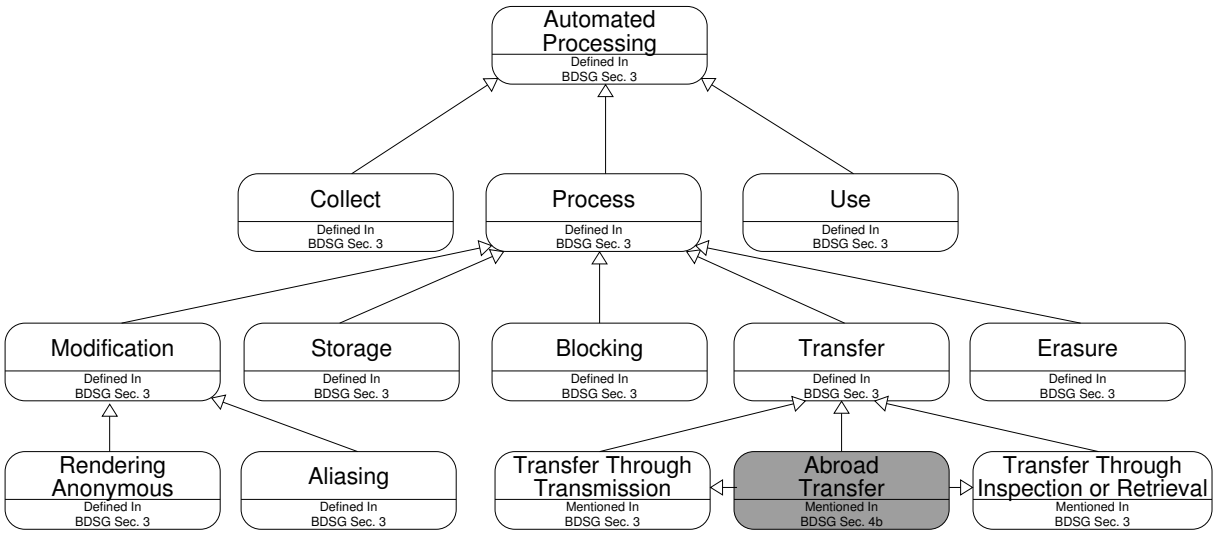


Figure 15.4.: *Updated law structure element hierarchy for activities after adding activities metioned in BDSG Section 4b*

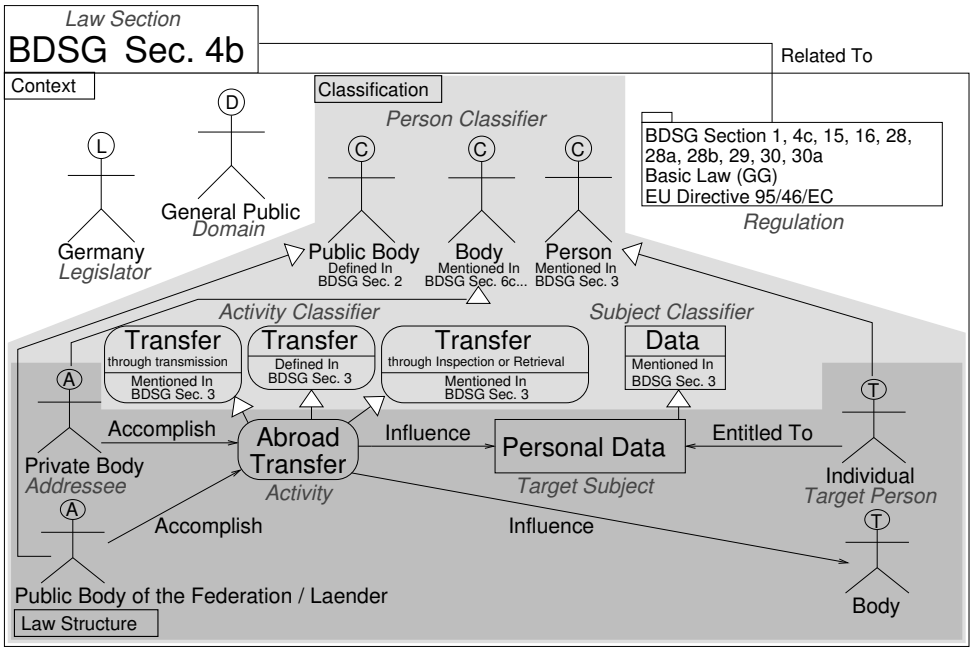


Figure 15.5.: *Law Pattern Instance for BDSG Section 4b*

Running Example:
The resulting Law Pattern instance for BDSG section 4b is shown in Fig. 15.5. It reflects all the information collected up to this step. In the white *context part* it shows the *legislator Germany* and the *domain General Public*. It also shows all related regulations containing *section 1* (Note, that Fig. 15.5 already shows the final Law Pattern instance after the federal data protection act is fully modeled. Thus, there are some related regulations defined by dictates of justice which follow after the current complete dictate of justice.). The dark gray *law structure* part contains the *addressees public body of the federation, public body of the Länder, private body*. The addressees accomplish the *activity abroad transfer* which influences the *target subject personal data* and the *target person body*. The *target person individual* is entitled to the personal data. The light gray *classification* part shows the parent law structure elements for the law structure elements contained in the law structure. This part is instantiated using the law structure element hierarchies. For example, Fig. 15.4 shows the law structure element hierarchy for activities. This hierarchy tells us that abroad transfer is a refinement of the *activity classifiers transfer, transfer through transmission, transfer through inspection or retrieval*. Hence, we add these three activity classifiers to the classification part. In the same way we add the according *person classifiers body, public body, and person*, and the *subject classifier data*. Note, that the target person body is a root element in the according law structure element hierarchy for persons. Thus, it does not add an element to the classification part.

15.1.4. Instantiation of a Referring Dictate of Justice

Referring dictates of justice not only contain references within one law but also define references to other laws. Thus, we have to *identify the referenced law and section* and check whether the law, which is referenced by the referring dictate of justice, is already modeled or not. In case it is not already modeled, we have to *add the law* and the particular section which is referenced. The newly added law might be a candidate to be fully modeled in the future.

Running Example:
Table 15.3 shows the referring dictate of justice contained in section 4b. It contains several references. The *referenced law* is the *BDSG* itself. The *referenced sections* are 15, 16, and 28 to 30. We do not have to model any new law or section as we stay within the BDSG.

Then we *identify the connected complete dictate of justice*. A referring dictate always adds further information to a complete dictate of justice (*the connected dictate of justice*). This might be information about competing regulations, which have to be discussed, complementing regulations, which have to be obeyed at the same time as the connected dictate of justice, substitutional dictates of justice which can be obeyed instead of the connected dictate of justice, or just a further dictate of justice which contains additional law structure elements or legal consequences. The referred law and section, the dictate of justice at hand and the connected dictate of justice are related regulations. Hence, we have to *update* the related regulations for the *connected dictate of justice*.

Running Example:
This particular referring dictate of justice is defined in the direct context of its connected dictate of justice. It is the complete dictate of justice contained in section 4b of the federal data protection act. Hence, we update the related regulations of this complete dictate of justice. Figure 15.5 already reflects the referring dictate of justice in the related regulations in the white context part.

15.1.5. Instantiation of a Restricting Dictate of Justice

For adding a restricting dictate of justice, we *identify the restricted complete dictate of justice*. A restricting dictate of justice does not define a law structure, but adds restrictions to the core

(1) *In connection with activities which fall in part or in their entirety within the scope of the law of the European Communities, the transfer of personal data to bodies other than those stated in* Section 4b (1) ***(restricted dictate)*** *above shall be admissible even if such bodies do not guarantee an adequate level of data protection, in so far as*

1. *the data subject has given his/her consent,*
2. *the transfer is necessary for the performance of a contract between the data subject and the controller or the implementation of pre-contractual measures taken in response to the data subject's request,*
3. *the transfer is necessary for the conclusion or performance of a contract which has been or is to be entered into in the interest of the data subject between the controller and a third party,*
4. *the transfer is necessary on important public interest grounds, or for the establishment, exercise or defense of legal claims,*
5. *the transfer is necessary in order to protect the vital interests of the data subject,*
6. *the transfer is made from a register which is intended to provide information to the public and which is open to consultation either by the public in general or by any person who can demonstrate a legitimate interest, to the extent that the statutory conditions are fulfilled in the particular case.*

It shall be pointed out to the recipient body that the transferred data may be processed or used only for the purpose for which they have been transferred. (**Restricting Dictate of Justice**)
...

Table 15.4.: *Restricting Dictate Of Justice, contained in BDSG Section 4c (Bundestag der Bundesrepublik Deutschland (Lower House of German Parliament), 2009 [88])*

structure of a complete dictate of justice. Sometimes it also adds additional legal consequences. The *restricted complete dictate justice* is explicitly mentioned in the dictate of justice at hand or it is the last complete dictate of justice defined in the preceding sections. The restriction of the law structure or the additional legal consequence are not of relevance for finding relevant laws, but are of relevance for the legal revision and adjustment of requirements. Hence, we have to *update* the related regulations for the *restricted dictate of justice* as the restricted dictate and the dictate of justice at hand are related regulations.

Running Example:
Table 15.4 shows a restricting dictate of justice contained in section 4c of the BDSG. It mentions *section 4b* as restricted section. Thus, the *restricted complete dictate of justice* is the one contained in section 4b (see Table 15.3). Hence, we update this complete dictate of justice accordingly and add section 4c as related section for the complete dictate of justice contained in section 4b. Figure 15.5 already reflects the referring dictate of justice in the related regulations in the white context part.

15.2. Preparing Transformation Cards

Another means, beside the Law Pattern instances, which can be prepared once and then be reused for every execution of the law identification process, are the so called transformation cards. We propose transformation cards to support the elicitation of additional context which is important to know for the identification of laws, and the transformation of requirements directly into Law Identification Pattern instances. These transformation cards can be applied whenever the functional requirements of a system to be are already expressed as problem diagrams, and a transformation card is applicable for a specific problem frame and consists of the parts *matching*, *legal context questions*, and *transformation*.

A transformation card supports and guides requirements engineers when preparing the requirements for matching with relevant laws. In this way, the identification of laws gains precision and is less error-prone, for example due to forgetting important domain knowledge. Beside improving the precision and reducing the chance of an error, transformation cards are the basis for the semi-automatic tool-support. The description of the tool-support will also be given in the following paragraphs as follows:

Tool:
For the tool-support, all information given by the transformation cards is modeled using UML and a specific transformation card profile. The model containing all transformation cards is then the input for the tool. A detailed discussion of the architecture of the tool and the technologies used is available in Chapter 21[16]

For the *matching* of a problem diagram and this transformation card, one has to check if the *referred and constrained domains* are of the *type* as given in the matching part.

[16] Page 353

Problem Frame: Required Behavior

Matching

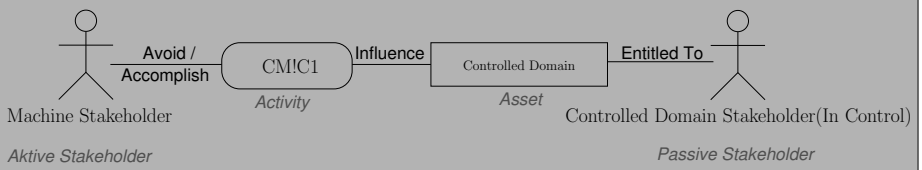
Referred domain type(s)	Constrained domain type(s)	Comment	Sequence of Phenomena
-	C	-	([CD!C2];CM!C1;[CD!C2])*

Legal Context Questions

Necessary Information	Details	Question	Result (Modeling Rules)
Machine Stakeholder	-	Who is responsible for and in control of the machine?	**For each** found stakeholder : **If** stakeholder does not exist in model **then Add** new ≪biddable≫ domain for the stakeholder to the model.**fI Add** a ≪controls≫ association between stakeholder domain and machine.**roF**
Controlled Domain Stakeholder(In Control)	-	Who is responsible and in control of the controlled domain?	**For each** found stakeholder : **If** stakeholder does not exist in model **then Add** new ≪biddable≫ domain for the stakeholder to the model.**fI Add** a ≪controls≫ association between stakeholder domain and controlled domain.**roF**
Controlled Domain Stakeholder(Influenced)	-	Who is influenced by the controlled domain?	**For each** found stakeholder : **If** stakeholder does not exist in model **then Add** new ≪biddable≫ domain for the stakeholder to the model.**fI Add** a ≪influenced≫ association between stakeholder domain and controlled domain.**roF**
Controlled Domain Stakeholder(Observed)	-	Who is observed by the controlled domain?	**For each** found stakeholder : **If** stakeholder does not exist in model **then Add** new ≪biddable≫ domain for the stakeholder to the model.**fI Add** a ≪entitledTo≫ association between stakeholder domain and controlled domain.**roF**

Transformation

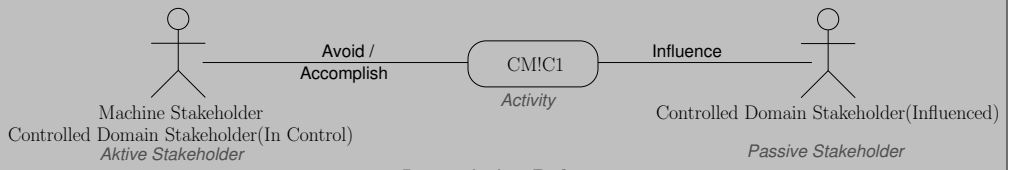
Core Structure Variant 1



Instantiation Rule:

For each machine stakeholder:
For each controlled domain stakeholder(In Control): Instantiate core structure variant 2
roF roF

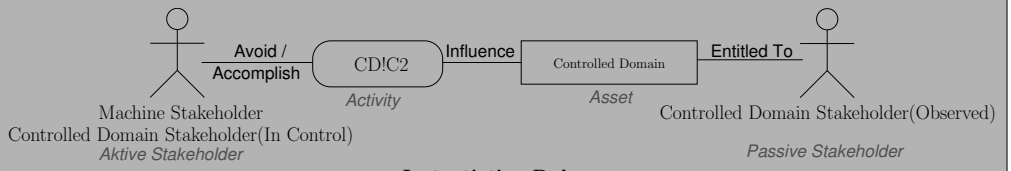
Core Structure Variant 2



Instantiation Rule:

For each machine stakeholder:
For each controlled domain stakeholder(in control):
For each controlled domain stakeholder(influenced):Instantiate core structure variant 2.
roF roF roF

Core Structure Variant 3



Instantiation Rule:

For each machine stakeholder:
For each controlled domain stakeholder(in control):
For each controlled domain stakeholder(observed): Instantiate core structure variant 2
roF roF roF

Table 15.5.: *Transformation Card: Required Behavior*

Running Example:
Table 15.5 shows the transformation card for the problem frame required behavior. In case of the required behavior frame the requirement does not refer to a domain, and constrains the controlled domain, which is a causal domain.

Additionally, the sequence of phenomena given as regular expression in the matching part of the transformation card, has to match the sequence of phenomena as given by the requirement of the problem diagram.

Running Example:
For the required behavior frame the machine must control a phenomenon which is shared with the controlled domain. Additionally, there might be phenomena which are controlled by the controlled domain. These phenomena might be issued by the controlled domain before and after the phenomena controlled by the control machine are issued.

The problem diagrams have sometimes to be modified for matching. For example, big and complex problem diagrams have to be partitioned, or domains have to be merged for fitting the problem diagram at hand to a problem frame. The interested reader is referred to Hatebur and Heisel (2010 [176]) on this matter. After checking the matching part of a transformation card, it is clear if it is applicable for the problem diagram at hand.

Tool:
For selecting the applicable transformation cards, the tool provides a selection of transformation cards which might be applicable to the problem. For this task we use the Epsilon comparison language. For each transformation card it calculates a fitting indicator for a given problem diagram. In case that all domains and necessary interfaces are found in a problem diagram, and the correct domains are referred and constrained. Hence, the fitting indicator is 1. The indicator is decreased for each missing domain, additional domain or differing interface. The selection of an applicable transformation card itself has to be done by the user. It cannot be done automatically because the sequence of phenomena, for example, cannot be checked automatically. But for the selection, the user has only to analyze the transformation cards with a high indicator and not all transformation cards.

In many cases, requirements do not contain all the information which is of importance for identifying relevant laws. For example, the actual stakeholders which are responsible for the actions the machine takes are hidden, especially when the machine automatically takes action without a direct command by a biddable domain. To improve this shortcoming (from a legal perspective) of requirements, each of our transformation cards also contains a questionnaire which helps to elicit this legal context. The *legal context questions* part of a transformation card contains several rows. Each row states the *necessary information* we are looking for, if this information *details* a domain we already know (for example, we collect information contained in a lexical domain already known), the *question* itself, and rules to model the *result* which is based on the answer to the question. For modeling the answers, we use domain knowledge diagrams (Alebrahim et al., 2011 [5]) and a UML profile for legal domain knowledge modeling. Figure 15.6 shows this profile. It contains four new stereotypes for modeling a ≪`behalfOf`≫, ≪`controls`≫, ≪`influences`≫ and ≪`entitledTo`≫ relation. These stereotypes are refinements of the already existing ≪`constrains`≫ stereotype, which is used for modeling problem

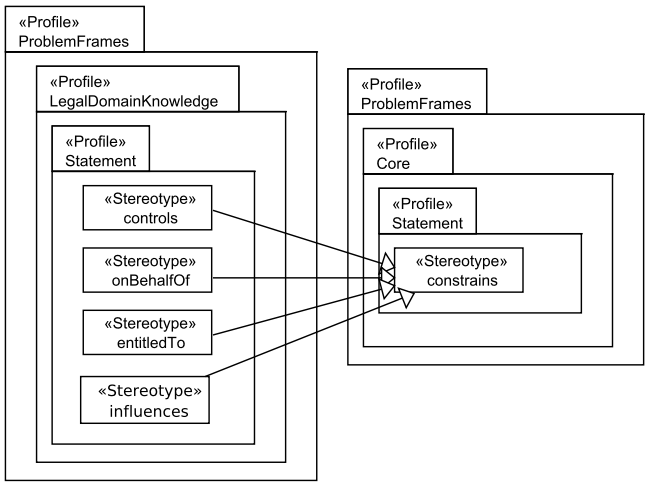


Figure 15.6.: *UML4PF Profile Extension for Supporting Modeling of Legal Domain Knowledge*

diagrams, problem frames and domain knowledge diagrams. All other information collected during the domain knowledge collection can be expressed using existing UML or UML4PF notational elements. For example, for the structure of lexical domains, the UML provides enough notion elements, like the composition and aggregation relation and properties, to express the structure.

Running Example:
For example, in the first row of the legal context questions part of the required behavior transformation card (see Table 15.5), the necessary information we are looking for is the machine stakeholder. The machine stakeholder is the one who controls the machine and is liable for its behavior. The machine stakeholder does not detail any domain which is already part of the required behavior problem frame. The question we have to answer is "Who is responsible for and in control of the machine?". According to the rules given by the result column, we have to add for each stakeholder who is not already modeled as biddable domain an according biddable domain. Between the domain representing a machine stakeholder and the machine we have to add an association typed with a ≪`controls`≫ stereotype.

Tool:
The tool uses the Epsilon generation language and pdflatex to prepare an adjusted questionnaire. Table 16.3[17] shows such a generated questionnaire. The shown questionnaire only contains some questions regarding one transformation card before the first iteration of answering questions. Indeed, the tool does not generate one questionnaire for each problem diagram and matching transformation card separately. Instead, it generates one coherent questionnaire for a complete set of problem diagrams and related transformation cards. This way, repeating questions can be treated once under the light of the problem diagrams which raised the question. Several iterations of answering questions are possible as an answer can lead to new questions. For example, discovering a new sub-part of a lexical domain leads to new questions regarding this new part. Thus, the tool indicates the need of an additional iteration and provides a new questionnaire. The new questionnaire only contains the newly occurring questions. In this way, the generated questionnaires guide through the domain knowledge elicitation. The modeling of domain knowledge is completely tool-supported. Using the Epsilon wizard language, the answers

to the questions can be added within wizards. The domain knowledge diagrams according to the answers themselves are then automatically set up. Only the graphical representation has to be generated manually if needed.

The *transformation* part of a transformation card defines the transformation of the problem diagram at hand and the according domain knowledge to different core structures of Law Identification Pattern instances. At this point it is important to note that one problem diagram might give rise to several core structures, because the requirement modeled using a problem diagram can contain several different cases from a legal perspective. In consequence, the transformation part contains several variants. For each variant there might be several instances with changing stakeholders. A variant defines the elements of the core structure relevant (given as figure), and an instantiation rule to populate the core structure.

Running Example:
For example, core structure variant 1 is used to generate core structures containing a machine stakeholder as active stakeholder, the phenomenon CM!C1 as activity, the controlled domain as asset, and a controlled domain stakeholder (in control) as passive stakeholder. We have to generate a core structure for each machine stakeholder and controlled domain stakeholder (in control) we identified when answering the questions.

Tool:
The transformation from problem diagrams to core structures can be performed fully automatically by the tool. For this task we use the Epsilon transformation language and a UML profile for representing law identification patterns in UML. Note that the core structures are added to the same UML model where also the problem diagrams reside. In this way, we have one model storing all information allowing information tracing and coherence checks. In a second step, the UML representation is transformed into a Law Pattern specific form, which allows the matching with laws. The tool-support as such is discussed in Chapter 21[18].

15.3. Conclusion

In this chapter we presented a detailed guidance on steps and means which prepare the law identification, but which are not conducted for every execution of the law identification process. Both, the modeling of laws and the preparation of transformation cards, require a significant amount of effort to be taken. But this effort pays off, because it has only to be invested once while the results are of use for many executions of the law identification process. Additionally, we provide the transformation cards for many problem frames. The contributions of this chapter are

- a detailed guidance for instantiating the several types of dictates of justice,
- the idea of transformation cards for easing the instantiation of Law Identification Patterns,
- a tool support for using the transformation cards.

[17] Page 286
[18] Page 353

Chapter 16

Legal Compliance Requirements Elicitation

Within this chapter, we explain the steps which have to be executed for every run of the law identification process as introduced in Section 14.5[1]. These steps make use of the prepared Law Pattern instances (Section 15.1[2]) and transformation cards (Section 15.2) we discussed in the previous chapter.

In Section 16.1, we explain how to structure requirements in such a way that they are suitable for finding laws using our Law Identification Pattern. This section is based on Faßbender and Heisel (2013 [133]), and Faßbender and Heisel (2014 [134]). After the preparation of the requirements, the gap between the legal and the requirements domain has to be bridged. How to do so is explained in Section 16.2[3]. This section is novel work firstly presented in this thesis. The same applies for Section 16.3[4], which shows how to match Law Pattern instances and Law Identification Pattern instances, and how to derive possibly relevant laws and dictates of justice this way. In Section 16.4[5], we briefly discuss the legal revision which identifies the actual relevant laws and dictates of justice. This step is based on the discussions and opinions of the legal experts. After the relevant laws and dictates of justice are known, we discuss how these laws and sections influence the according requirements in Section 16.5[6]. A first idea was already sketched in Beckers, Faßbender, and Schmidt (2012 [44])[7] for security requirements, but the general idea how to deal with the influence of laws on requirements is novel. With this step our proposed process is finished.

16.1. Instantiation of Law Identification Pattern (Core+Context)

In the following, we present a guided and tool-supported transformation of requirements into Law Identification Pattern instances. This section covers the activity *instantiation of Law Identification Patterns (core+context)* of the general law identification process (see Figure 14.12[8]). Again, we make use of the problem frames approach to structure the requirements in terms of problem diagrams in the first place. Note that the steps described here are specifically tailored to the Problem Frame notation, while the steps described for instantiating the Law Identification Pattern in the Context Pattern catalog (Chapter B.1.3[9]) are of a more general nature.

[1] Page 262
[2] Page 267
[3] Page 295
[4] Page 298
[5] Page 302
[6] Page 302
[7] The method for deriving security requirements was a contribution of Kristian Beckers.
[8] Page 263
[9] Page 413

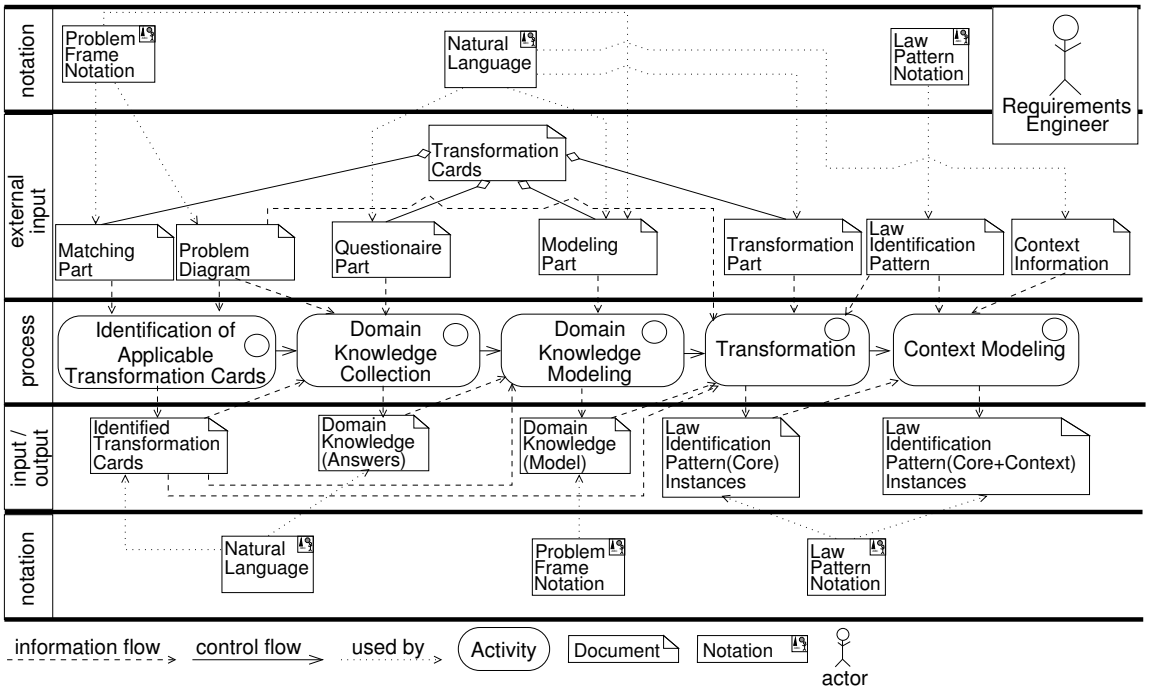


Figure 16.1.: *The transformation process*

Thus, both descriptions differ but the steps described in the following are a refinement of the more general steps described in the Law Identification Pattern description. Beside the fact that we use problem frames in all other steps within this thesis, there are several other important reasons for using problem frames. We decided to use problem frames because they have a kind of semi-formal structure and can be modeled. Furthermore, they already embody descriptions of common problems. Thus, they are suitable as an input for a transformation as they have a predictable structure, and transformation rules can be set up on the basis of the generic problems. We will show how to turn problem diagrams into law identification patterns using our tool. We provide detailed transformation rules for different problem frames (the enumeration of problem frames considered by us can be found in Côté, Hatebur, Heisel, Schmidt, and Wentzlaff (2008 [103])), to obtain the corresponding Law Identification Pattern instances.

The transformation cards as introduced in the previous chapter are the central tool for executing the transformation in the following. A transformation card contains information used for *matching problem diagrams and frames*, and information how the problem frame, and therefore the matching problem diagram, is related to the *core structure* of the law identification pattern. It also contains information for *collecting potentially missing domain knowledge*, which is important for transforming problem diagrams into core structures, and the *transformation rules* themselves. As a result, a transformation card supports and guides requirements engineers when preparing the requirements for the matching with relevant laws. In this way, the identification of laws gains precision and is less error-prone, for example due to forgetting important domain knowledge. Beside improving the precision and reducing the chance of an error, transformation cards are the basis for our semi-automatic tool-support.

The process for using the transformation cards, and therefore the tool, is shown in Fig. 16.1. It starts with the *identification of applicable transformation cards* for the problem diagrams. The *problem diagrams* are a necessary external input like the *matching part* of the transformation cards. The problem diagrams have to be modeled beforehand. Then, the *questionnaire part* of the *identified transformation cards* is used for a *domain knowledge collection*. Next, the *answers*

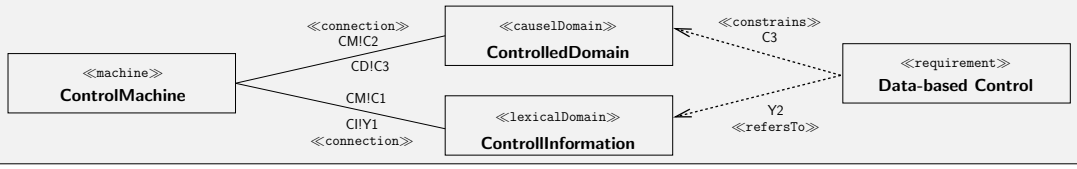
and the *modeling part* of the identified transformation card serve as input for *domain knowledge modeling*. The modeling part contains detailed rules how to model legal domain knowledge. The *transformation* is executed using information contained in problem diagrams and the according *domain knowledge models*. The used transformation rules are obtained from the *transformation part* of the identified transformation cards. The transformation results in *law identification pattern (core) instances*. Last, the *context modeling* takes place. The already elicited context for the system-to-be and the requirements is added to the *Law Identification Pattern (core) instances*. In particular, this includes information about the general context as described and elicited in Chapter 10[10], and the relations between processes and requirements as described and elicited in Chapter 11[11] and Chapter 12[12]. Finally, we get the *Law Identification Pattern (core + context) instances.*

16.1.1. Identification of Applicable Transformation Cards

Table 16.1 shows the matching part of the transformation card for the data-based control problem frame. Now, we have to check if the transformation card is applicable or not. The first thing to be checked is if the domains which are referred or constrained have the correct type as described in the matching part.

[10]Page 181
[11]Page 203
[12]Page 213

Problem Frame: Data-Based Control			
Matching			
Referred domain type(s)	**Constrained domain type(s)**	**Comment**	**Sequence of Phenomena**
X	C		[CM!C1 | CI!Y1; | CM!C1; CI!Y1;];CM!C2;[CD!C3;]



. . .

Table 16.1.: *Data-Based Control: Matching Part*

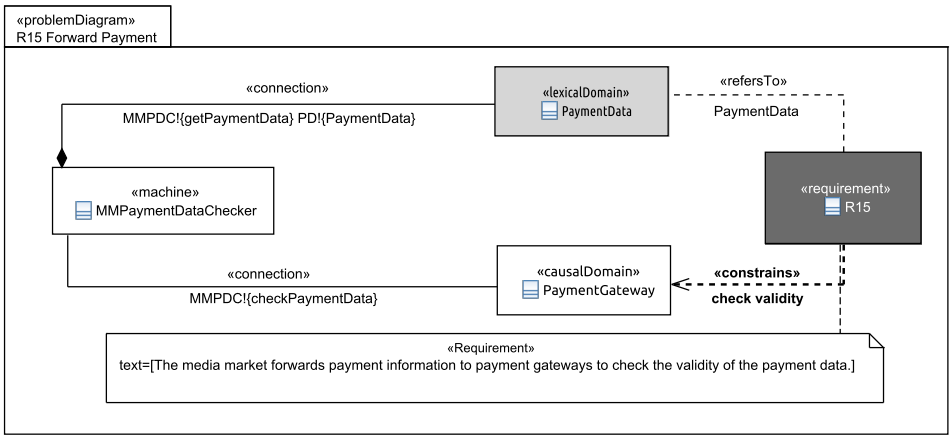


Figure 16.2.: *Problem Diagram for R15*

Running Example:
As an example we will use `R15` as depicted in Fig. 16.2. The **PaymentData** is a lexical domain. It is referred to. This matches the *Referred domain type(s)* of the matching part. This is also true for the *Constrained domain type(s)*, which has to be a causal domain. The constrained domain **PaymentGateway** in our problem diagram is a causal domain. Thus, this transformation card remains an applicable candidate.

Next, we have to check if the overall structure of the problem diagram matches the problem frame.

Running Example:
For the structure, the problem frame, as described by the graphical representation in the matching part, is exactly the same. For other cases it might be that there are, for example, additional domains. In such a case we would have to check if the additional domains can be merged with other domains to fit to the problem frame at hand.

Last, we have to check whether the implicitly described interaction of the problem frame, by means of the phenomena, matches the phenomena of the problem diagram or not.

Running Example:
The sequence of *MMPDC!*{*getPaymentData*}; *PD!*{*PaymentData*}; *MMPDC!*{*checkPaymentData*}; matches the regular expression as given by the *Sequence of Phenomena* in the transformation card. In this case, the sequence of phenomena [CM!C1 | CI!Y1; | CM!C1;CI!Y1;]; CM!C2;[CM!C3;] as described by the transformation card perfectly fits to the sequence of phenomena described. As a result, the transformation card for the problem frame data-based control has to be applied.

Problem Frame: Data-Based Control

...

Legal Context Questions			
Necessary Information	**Details**	**Question**	**Result (Modeling Rules)**
Structure Control-Information	Control-Information	Which information is contained in the control information and which structure does it have?	**Add** a ≪`domainKnowledge`≫ element. **Add** a ≪`constrains`≫ dependency from the ≪`domainKnowledge`≫ element to the control information **For each** new found part of the control information **:** **If** a domain for control information part does not exist **then Add** a ≪`lexical`≫ domain for the control information part.**endIf** **Add** a aggregation relation between the control information domain and the control information part domain. **Add** a ≪`refersTo`≫ dependency from the ≪`domainKnowledge`≫ element to the (newly added) ≪`lexical`≫ domain.**roF**

Table 16.2 – continued on next page

Table 16.2 – continued from previous page

Legal Context Questions			
Necessary Information	**Details**	**Question**	**Result**
Control-Information Stakeholder	-	Who are the stakeholders owning the storage containing the ControlInformation?	**Add** a ≪`domainKnowledge`≫ element. **Add** a ≪`controls`≫ dependency from the ≪`domainKnowledge`≫ element to the control information **For each** found stakeholder **:** **If** stakeholder does not exist in model **then Add** new ≪`biddable`≫ domain for the stakeholder to the model.**endIf** **Add** a ≪`refersTo`≫ dependency from the ≪`domainKnowledge`≫ element to the (newly added) ≪`biddable`≫ domain.**roF**
Control-Information Information Stakeholder	-	Who are the stakeholders entitled to information contained in the control information (part)?	**For each** found stakeholder **:** **Add** a ≪`domainKnowledge`≫ element. **If** stakeholder does not exist in model **then Add** new ≪`biddable`≫ domain for the stakeholder to the model.**endIf** **Add** a ≪`refersTo`≫ dependency from the ≪`domainKnowledge`≫ element to the (newly added) ≪`biddable`≫ domain. **For each** control information (part) to which the stakeholder is entitled: **Add** a ≪`entitledTo`≫ dependency from the ≪`domainKnowledge`≫ element to the control information (part) **roF roF**
Machine Stakeholder	-	Who is responsible and in control of the machine?	**Add** a ≪`domainKnowledge`≫ element. **Add** a ≪`controls`≫ dependency from the ≪`domainKnowledge`≫ element to the machine **For each** found stakeholder **:** **If** stakeholder does not exist in model **then Add** new ≪`biddable`≫ domain for the stakeholder to the model.**endIf** **Add** a ≪`refersTo`≫ dependency from the ≪`domainKnowledge`≫ element to the (newly added) ≪`biddable`≫ domain.**roF**
Controlled-Domain Stakeholder	-	Who is responsible and in control of the controlled domain?	**Add** a ≪`domainKnowledge`≫ element. **Add** a ≪`controls`≫ dependency from the ≪`domainKnowledge`≫ element to the controlled domain **For each** found stakeholder **:** **If** stakeholder does not exist in model **then Add** new ≪`biddable`≫ domain for the stakeholder to the model.**endIf** **Add** a ≪`refersTo`≫ dependency from the ≪`domainKnowledge`≫ element to the (newly added) ≪`biddable`≫ domain.**roF**
...			

...

Table 16.2.: *Data-Based Control Transformation Card: Legal Context Questions Part*

16.1.2. Domain Knowledge Collection

After the successful matching, the transformation card contains further guidance for preparing the transformation. The different core structure variants as given in the transformation part do not only relate a problem frame and core structure, but also consider typical domain knowledge for a problem frame. To ensure that this domain knowledge is collected properly, there is a questionnaire which gives guidance for collecting the domain knowledge. While answering these questions, necessary domain knowledge which might be missing is collected (The actual modeling of the collected information is explained in Section 16.1.3).

Running Example:
Table 16.2 shows some of the questions for the data-based control transformation card. There

are more questions which are not of relevance in the following. Table 16.3 shows the questions contained in Table 16.2 adjusted to the problem diagram shown in Fig. 16.2.

The first information, which might be missing, is about the structure of the **ControlInformation** (Question: "Which information is contained in the control information and which structure does it have?"). Normally, in problem diagrams a lexical domain represents information relevant for the problem at hand. This particular information is often modeled as part of an overall database or hidden in an information which aggregates different pieces of information. But for law identification we need to know the specific piece(s) of information which is/are relevant for the problem at hand. In case the database or aggregated information can be partitioned further, we have to add a separate lexical domain for each found piece of information.

Running Example:
For the **PaymentData** we collect additional information about its structure. Afterward, we know that personal customer information, for example his/her full name is part of the payment data. This name is related to transaction data including, for example, the bank account identifier, authorization data and amount of money. Additionally, the information about the bank is stored in the payment data. Thus, we add the these information pieces as part of the payment data.

Another kind of information, which might not have been modeled, is about the stakeholder in control of a lexical domain (Question: "Who are the stakeholders owning the storage containing the ControlInformation?"). In case of the data-based control problem frame this is the **ControlInformation**. This particular information is not important for fulfilling any requirements. But for a legal analysis it is often of central relevance who is in control of a certain means like a database, because such a stakeholder might influence the means, and might have rights but also obligations regarding the means.

Running Example:
The payment data is related to one **ControlInformation Stakeholder**. This is the **ContentAggregator**, who owns the media market and therefore the storage of the payment data.

Beside the stakeholder controlling a particular information, it is also often of relevance which other stakeholders have rights regarding the information contained in a lexical domain even

Legal Context Questions		
Necessary Information	**Details**	**Question**
Structure PaymentData	PaymentData	Which information is contained in the payment data and which structure does it have?
PaymentData Stakeholder	-	Who are the stakeholders owning the storage containing PaymentData?
PaymentData Information Stakeholder	-	Who are the stakeholders entitled to information contained in the PaymentData?
MMPaymentDataChecker Stakeholder	-	Who is responsible for and in control of the MMPaymentDataChecker?
PaymentGateway Stakeholder	-	Who is responsible for and in control of the PaymentGateway?
...		

Table 16.3.: *Catalog of questions (Part)*

though they do not directly control the lexical domain (Question: "Who are the stakeholders entitled to information contained in the control information (part)?"). Such stakeholders might be entitled to the information, because there is, for example, a contractual relationship granting them rights or universal rights like authorship or data protection. Note that for this question all stakeholders are collected which might have a right or influence on the control information. It is not part of the question to already judge whether there is a legal justification for the assumed rights or not. Thus, we collect all stakeholders which contribute to forming the control information.

Running Example:
The payment data and its parts are related to two PaymentData information stakeholders. The first one is the **Customer** because the customer information part of the payment data contains information about her / him and he / she has entered the transaction data. The second one is the **Bank** because the payment data contains information about the bank.

From a legal perspective, a machine or causal domain is always seen as intermediate means which executes the actions of a person (Person in this case does not only include real persons but also, for example, legal persons such as companies.). Therefore, for judging a legal case, every action taken by a machine or causal domain is treated like it is performed by the one responsible and in control of the machine or causal domain (Questions: "Who is responsible and in control of the machine?", "Who is responsible and in control of the controlled domain?).

Running Example:
As the media market, and therefore also the machine **MMPaymentDataChecker**, is owned by the **ContentAggregator**, the **Machine Stakeholder** is the content aggregator. The **PaymentGateway** is owned by the **PaymentGatewayProvider** which in turn is the **Controlled-Domain Stakeholder**.

Another important information for judging the relevance of certain laws is the case where one person is acting on behalf of another person. In such a case activities which are not permitted for the person who is acting as proxy might be permitted for the other person. In this case the activity might also be permitted for the proxy person as long as it is allowed to delegate such an activity and there is a juridical proof that the proxy person is acting on behalf of the other person.

Running Example:
As the problem frame data-based control does not contain a biddable domain, there is no question raised regarding persons acting on behalf of other persons.

After answering the questionnaire, all necessary domain knowledge is available in terms of natural language answers. Note that often a question regarding a certain domain is asked several times. The reason is that domains are used in different problem diagrams. Each problem diagram defines a different context the domain is used in. Hence, the answers given to a question might differ from problem diagram to problem diagram, as the one who is answering is focusing on different aspects of the domain. For example, for the structure and parts of a lexical domain the answers might only focus on the parts relevant for the problem diagram at hand. But there might be parts which are important for other problem diagrams. Hence, the questions are

repeated for each problem diagram. In this way, we ensure the complete view on the relevant legal domain knowledge. For the modeling, which is explained in the next section, answers which are given more than once can be aggregated and only have to be modeled once.

16.1.3. Domain Knowledge Modeling

Each transformation card not only contains the questions for collecting the necessary domain knowledge but also instructions how to model them[13] (Column "Result (Modeling Rules)" in Table 16.2). In general, for modeling the necessary domain knowledge, we first have to decide if the newly obtained knowledge is an assumption, definition or designation, or a fact. This decision has to be taken for each obtained information individually. If an information is universally true (a fact), if we only assume it (assumption), or if we define it (definition), depends on the context in which we collect the information and the kind of knowledge we can rely on. Thus, it is not possible to decide beforehand which domain knowledge type has to be used for modeling, for example, a ≪`onBehalf`≫-relation. In consequence, the modeling rules in the "Result (Modeling Rules)"- column advise to add a domain knowledge element, while the modeler has to replace the general domain knowledge with fact, assumption or definition. For all kinds of domain knowledge, we have to add a domain knowledge diagram, which contains a model representing the domain knowledge.

The first type of information we collected in the previous step is about the structure and parts of lexical domains. The modeling steps to be taken are described in the "Results (Modeling Rules)"-column in Table 16.2. First, we have to **add** a domain knowledge (fact, assumption or definition) element and **add** a ≪`constrains`≫ dependency from the domain knowledge element to the ≪`lexicalDomain`≫ to be detailed. **For each** of the information parts found, we have to check **if** the part is already modeled. If it is not modeled, **then** we have to **add** a ≪`lexicalDomain`≫ which represents this part. Next, we have to **add** an aggregation relation between the original lexical domain, which was detailed by the current question, and the new found part. Last, we have to **add** a ≪`refersTo`≫ dependency from the domain knowledge element to the newly added ≪`lexicalDomain`≫.

Running Example:
Figure 16.3 shows the domain knowledge diagram for the *structure of the payment data*. We

[13] Note that we will describe the modeling by hand to ease the understanding of the reader. By now, the modeling is done by the tool in the background.

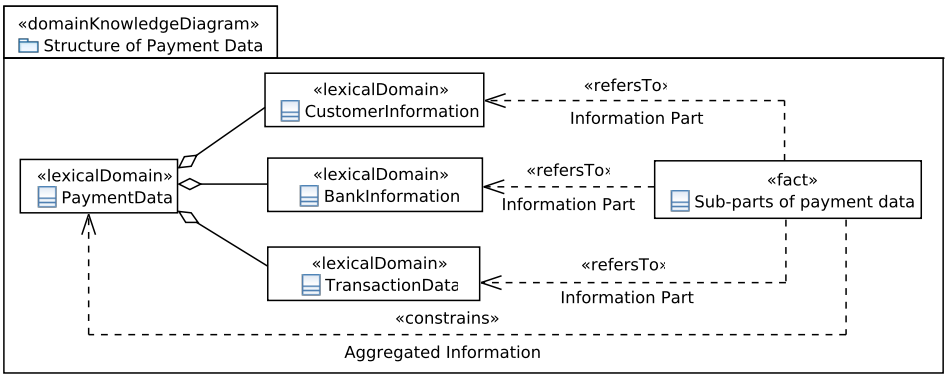


Figure 16.3.: *Domain Knowledge Diagram for the Structure of the Payment Data*

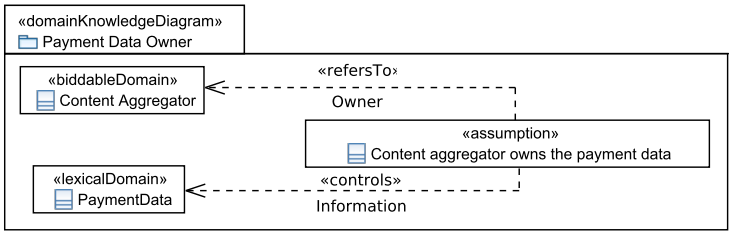


Figure 16.4.: *Domain Knowledge Diagram for the Owner of the Payment Data*

added a ≪`fact`≫ domain knowledge element, as there exists a standardized specification for the protocol and the payment data for validating payments. We also have added three ≪`lexicalDomain`≫s for the three new information parts **CustomerInformation**, **TransactionData**, and **BankInformation**. All these domains are related to the **PaymentData** using aggregation relations. The fact **Sub-parts of payment data** *refers to* the newly added domains and *constrains* the **PaymentData**.

The second type of information we collected in the previous step is about the stakeholder in control of certain lexical domains. The modeling steps to be taken are described in the "Results"-column in Table 16.2. First, we have to **add** a domain knowledge (fact, assumption or definition) element and **add** a ≪`controls`≫ dependency from the domain knowledge element to the ≪`lexicalDomain`≫ in question. **For each** of the stakeholders found, we have to check **if** the stakeholder is already modeled. If he / she is not modeled, **then** we have to **add** a ≪`biddableDomain`≫ which represents this stakeholder. Next, we have to **add** a ≪`refersTo`≫ dependency from the domain knowledge element to the (newly added) ≪`biddableDomain`≫.

Running Example:
Figure 16.4 shows the domain knowledge diagram for the *owner of the payment data*. We added an ≪`assumption`≫ domain knowledge element, as we only assume that whatever mean is used to store the data, this mean is under control of the **ContentAggregator**. As the **ContentAggregator** is already known, we do not have to model him / her again. The assumption **Content aggregator owns the payment data** *refers to* the **ContentAggregator** and *constrains* (*controls*) the **PaymentData**

The third type of information we collected in the previous step is about the stakeholders entitled to certain lexical domains. The modeling steps to be taken are described in the "Results"-column in Table 16.2. **For each** found stakeholder, we have to **add** a domain knowledge (fact, assumption or definition) element. If the stakeholder is not modeled, **then** we have to **add** a ≪`biddableDomain`≫ which represents this stakeholder. Next, we have to **add** a ≪`refersTo`≫ dependency from the domain knowledge element to the (newly added) ≪`biddableDomain`≫. **For each** information (part) to which the stakeholder is entitled, we have to **add** a ≪`refersTo`≫ dependency from the domain knowledge element to the ≪`lexicalDomain`≫.

Running Example:
Figure 16.5 shows the domain knowledge diagram for the *payment data information stakeholder* ***Bank***. We added an ≪`assumption`≫ domain knowledge element, as we only assume that the

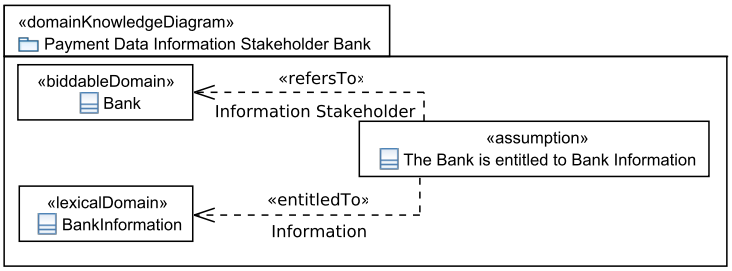


Figure 16.5.: *Domain Knowledge Diagram for the Payment Data Information Stakeholder Bank*

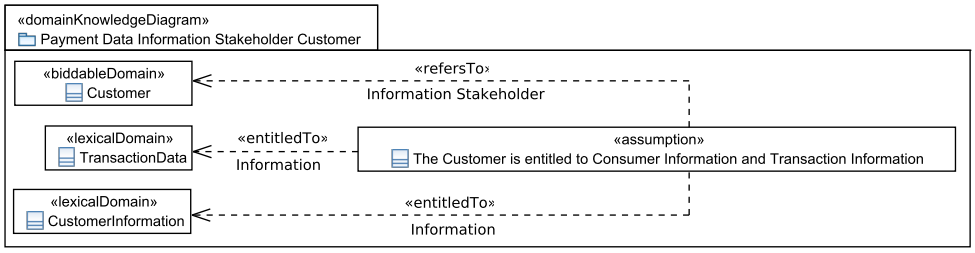


Figure 16.6.: *Domain Knowledge Diagram for the Payment Data Information Stakeholder Customer*

Bank is entitled to **BankInformation**. As the **Bank** is already known, we do not have to model it again. The assumption *refers to* the **Bank** and *constrains* (*entitledTo*) the **BankInformation**. Figure 16.6 shows the domain knowledge diagram for the *payment data information stakeholder* ***Customer***. We added an ≪`assumption`≫ domain knowledge element, as we only assume that the **Customer** is entitled to the **TransactionData** and to the **CustomerInformation**. It is only an assumption as the customer might not enter correct information. For example, the customer can also enter fake data or information about somebody else. As the **Customer** is already known, we do not have to model him / her again. The assumption *refers to* the **Customer** and *constrains* (*entitledTo*) the **TransactionData** and the **CustomerInformation**.

The fourth type of information we collected in the previous step is about the stakeholder in control of a machine or causal domain. The modeling steps to be taken are described in the "Results"-column in Table 16.2. First, we have to **add** a domain knowledge (fact, assumption or definition) element and **add** a ≪`controls`≫ dependency from the domain knowledge element to the ≪`machine`≫ / ≪`causalDomain`≫ in question. **For each** of the stakeholders found, we have to check **if** the stakeholder is already modeled. If he / she is not modeled, **then** we have to **add** a ≪`biddableDomain`≫ which represents this stakeholder. Next, we have to **add** a ≪`refersTo`≫ dependency from the domain knowledge element to the (newly added) ≪`biddableDomain`≫.

Running Example:
Figure 16.7 shows the domain knowledge diagram for the *owner of the MMPaymentDataChecker*. We add an ≪`assumption`≫ domain knowledge element, as we only assume that the machine is under full control of the **ContentAggregator**. As the **ContentAggregator** is already known, we do not have to model him / her again. The assumption **Content aggregator owns the payment data** *refers to* the **ContentAggregator** and *constrains* (*controls*) the

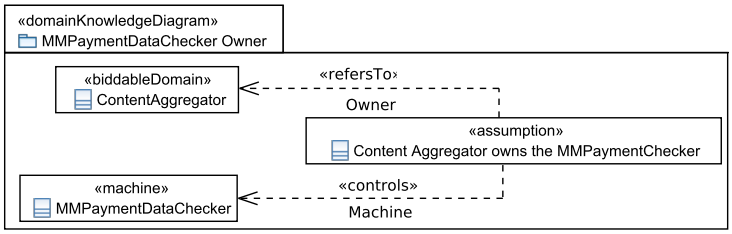


Figure 16.7.: *Domain Knowledge Diagram for the Owner of the MMPaymentDataChecker*

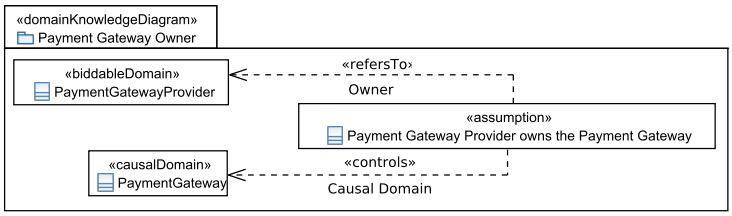


Figure 16.8.: *Domain Knowledge Diagram for the Owner of the Payment Gateway*

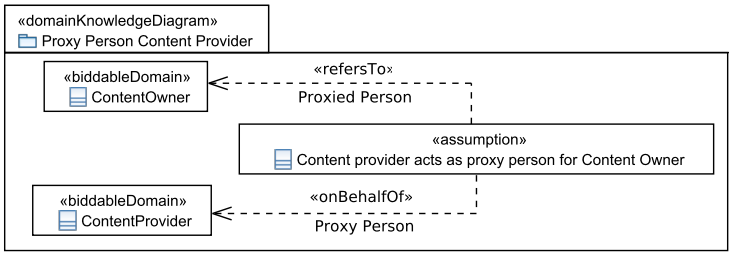


Figure 16.9.: *Domain Knowledge Diagram about the Content Provider As Proxy for Content Owners*

MMPaymentDataChecker. Figure 16.8 shows the domain knowledge diagram for the *owner of the PaymentGateway*. We add an ≪`assumption`≫ domain knowledge element, as we only assume that the **PaymentGateway** is under full control of the **PaymentGatewayProvider**. As the **PaymentGatewayProvider** is already known, we do not have to model him / her again. The assumption **Payment gateway provider owns the payment gateway** *refers to* the **PaymentGatewayProvider** and *constrains* (*controls*) the **PaymentGateway**.

The last type of information we collected in the previous step is about stakeholders acting on behalf of other stakeholders. For modeling this information, we have to **add** a domain knowledge (fact, assumption or definition) element and **add** an ≪`onBehalfOf`≫ dependency from the domain knowledge element to the ≪`biddableDomain`≫ acting as proxy. **For each** of the stakeholders for which he / she acts a proxy, we have to check **if** the stakeholder is already modeled. If he / she is not modeled, **then** we have to **add** a ≪`biddableDomain`≫ which represents this stakeholder. Next, we have to **add** a ≪`refersTo`≫ dependency from the domain knowledge element to the (newly added) ≪`biddableDomain`≫.

Running Example:
Figure 16.9 shows the domain knowledge diagram for the *content provider acting as proxy person*

for content owner which was collected for another problem diagram. We add an ≪`assumption`≫ domain knowledge element, as we only assume that **ContentProvider** executes all actions regarding the content on behalf of the actual **ContentOwner**. As the **ContentOwner** is not initially known, we do have to model him / her as ≪`biddableDomain`≫. The assumption **Content provider acts as proxy person for content owner** *refers to* the **ContentAggregator** and *constrains* (*onBehalfOf*) the **ContentProvider**

16.1.4. Transformation

With the newly obtained domain knowledge at hand, we can transform the problem diagram into Law Identification Pattern core structure instances. How often a core structure has to be instantiated is described in the instantiation rule (see Tab. 16.4) of each core structure variant given by the transformation part of a transformation card. Hence, it is possible that one problem diagram is transformed into several law identification pattern instances.

Table 16.4 shows the core structure variant for the data-based control transformation card. This variant focuses on the **machine stakeholder**, who is the *active stakeholder* in this core structure and who accomplishes the *activity* described by the phenomenon *CM!C1*. Note that we assume that each causal phenomenon is caused by a corresponding activity. This activity manifests in the environment in terms of the phenomenon CM!C1 which influences the **ControlInformation or a part of it**, which might be an *asset*. The **ControlInformation** or the **ControlInformation Information Stakeholder** are entitled to the asset and in turn to the *passive stakeholders*.

For `R15`[14] (Fig. 16.2[15]) and *Core Structure Variant 2* (Tab. 16.4) we have to instantiate the core structure **for each** combination of control information (part), machine stakeholder, control

[14]Page 1
[15]Page 283

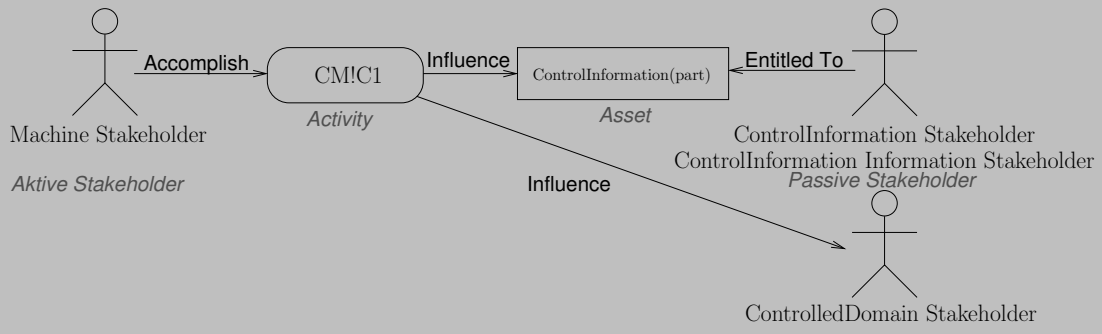
Problem Frame: Data-Based Control

...

...

Transformation

Core Structure Variant 2



Instantiation Rule:

For each machine stakeholder:
For each controlled domain stakeholder:
For each control information and its parts:
For each control information (information) stakeholder: Instantiate core structure variant 2. **roF roF roF**

...

Table 16.4.: *Data-Based Control Transformation Card: Transformation Part*

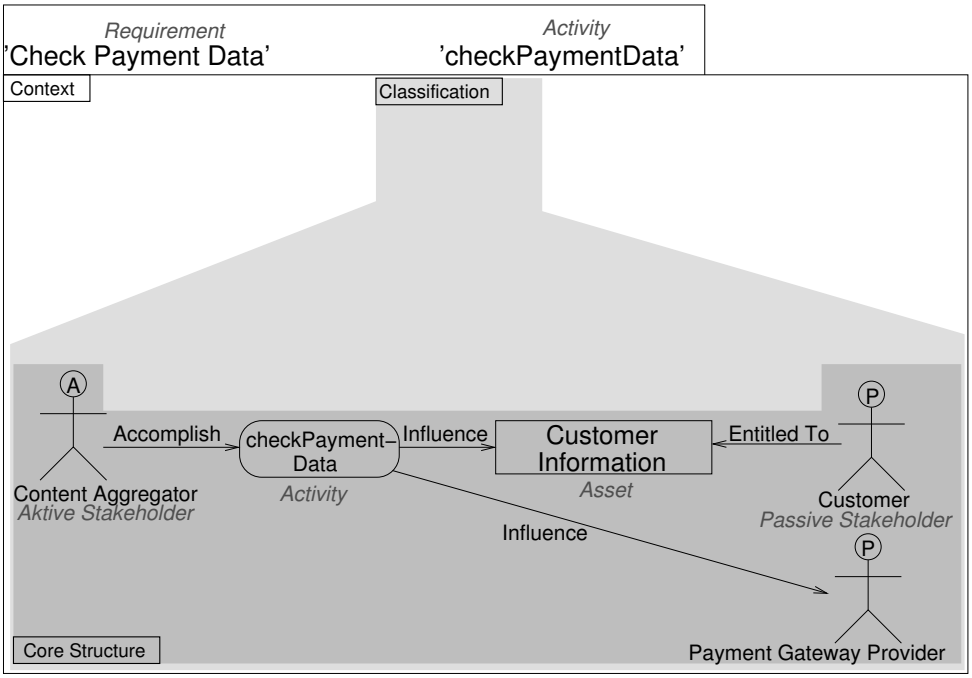


Figure 16.10.: *One Law Identification Pattern (Core) Instance (DBC_V2_3) for* `R15`

	Activity	*Active Stakeholder*	*Passive Stakeholder*	*Asset*	*Passive Stakeholder*	
Identifier	**Activity**	**Machine Stakeholder**	**ControlledDomain Stakeholder**	**ControlInformation (part)**	**ControlInformation Stakeholder**	**ControlInformation Information Stakeholder**
DBC_V2_1	checkPaymentData	ContentAggregator	PaymentGatewayProvider	PaymentData	ContentAggregator	
DBC_V2_2				CustomerInformation	ContentAggregator	
DBC_V2_3						Customer
DBC_V2_4				TransactionData	ContentAggregator	
DBC_V2_5						Customer
DBC_V2_6				BankInformation	ContentAggregator	
DBC_V2_7						Bank

Table 16.5.: *Instantiated Core Structures for* `R15` *and Core Structure Variant 2*

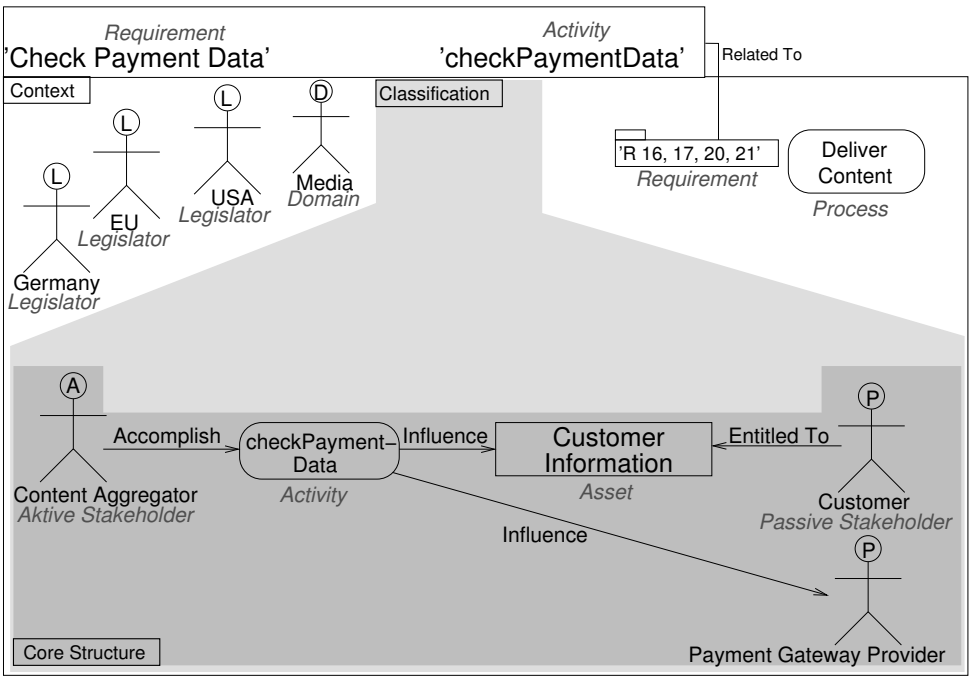


Figure 16.11.: *One Law Identification Pattern (Core+Context) Instance (DBC_V2_3) for* `R15`

information stakeholder or control information information stakeholder. The activity is the same for all core structures of variant 2.

Running Example:
This results in seven core structure instances as shown textually in Tab. 16.5. For all core structure instances the activity (*checkPaymentData*), the machine stakeholder (**ContentAggregator**), and the controlled domain stakeholder (**PaymentGatewayProvider**) remain the same. The first control information instance is the **PaymentData** itself. The control information stakeholder for the **PaymentData** is the **ContentAggregator**. There is no control information information stakeholder. Hence, we get one core structure instance as shown in the row with the identifier DBC_V2_1. The next control information part is the **CustomerInformation**. The control information stakeholder is **ContentAggregator**. This information is inferred from the **PaymentData** it is part of. The control information information stakeholder is the **Customer**. Hence, we get two more core structure instances as shown in the rows with the identifiers DBC_V2_2 and DBC_V2_3. Another control information part is the **TransactionData**. The control information stakeholder is **ContentAggregator**. The control information information stakeholder is the **Customer**. Hence, we get two additional core structure instances as shown in the rows with the identifiers DBC_V2_4 and DBC_V2_5. The last control information part is the **BankInformation**. The control information stakeholder is **ContentAggregator**. The control information information stakeholder is the **Bank**. Hence, we get the last two core structure instances as shown in the rows with the identifiers DBC_V2_6 and DBC_V2_7. An example of the graphical representation is shown in Fig. 16.10 for the law identification pattern core DBC_V2_3.

16.1.5. Context Modeling

The last information the requirements engineer can add alone without the assistants of a legal expert is about the context. Basically, in this step the requirements engineer has only to take parts of the already known information and add it to each Law Identification Pattern core. As the needed context information remains the same for all Law Identification Pattern instances related to one problem diagram, this information has to be collected only once for a problem diagram. The information about *legislators* and *domains* is already part of the *context pattern instance* as created in the context elicitation step described in Chapter 10[16]. The information about the *relation between requirements*, and therefore problem diagrams, *and processes* is known after executing the steps process elicitation (Section 11.2[17]) and functional requirements elicitation (Chapter 12[18]). The information about the *relation between requirements* is known after executing the steps functional requirements elicitation (Chapter 12[19]) and quality requirements elicitation (Chapter 13[20]).

Running Example:
From the already known SOA Stakeholder Pattern instance for the media market (Fig. 10.4[21]), we collect the *legislators* **Germany**, **EU**, and **USA**. Additionally, we collect the *domain* **Media**.

[16] Page 181
[17] Page 206
[18] Page 213
[19] Page 213
[20] Page 221

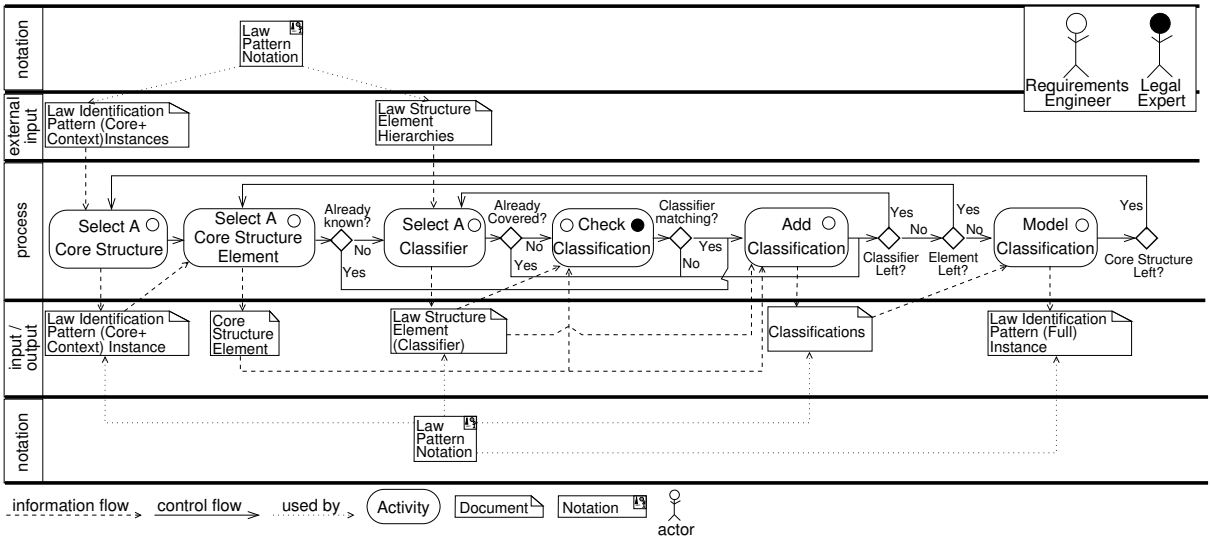


Figure 16.12.: *Process for Full Instantiation of Law Identification Pattern*

R15[22] is related to the process **deliver content** (Fig. 10.2[23]). The *related requirements* are **R19[24], R20[25], R23[26], and R24[27]**.

16.2. Full Instantiation of Law Identification Pattern

This section covers the activity *full instantiation of Law Identification Patterns* of the general law identification process (see Figure 14.12[28]). While a core structure can be instantiated by the requirements engineer alone, the full instantiation requires some legal expertise. Hence, legal experts have to be involved in the process of instantiating the full law identification pattern. Figure 16.12 shows this process.

The initial input to the first step of this process are the previously generated *Law Identification Pattern (core + context) instances*. We *select* one of the *Law Identification Pattern (core + context) instances*.

Running Example:
We select the Law Identification Pattern (core + context) instance as shown in Fig. 16.11. We assume that this is the first instance we are analyzing.

From this selected Law Identification Pattern core instance, we retrieve the *core structure elements* and *select* one. For the selected *core structure element* we check whether its classification

[21] Page 187
[22] Page 1
[23] Page 184
[24] Page 1
[25] Page 1
[26] Page 1
[27] Page 1
[28] Page 263

is *already known* or not. The more Law Identification Pattern core instances we have already fully instantiated, the more likely it is that a core structure element was already classified. The reason is, even though the number of Law Identification Pattern core instances can be very high, the number of possible core structure elements is limited by the environment of the machine and therefore this number remains stable. Thus, core structure elements are used several times in different law identification pattern core instances. In case that the core structure element at hand is already classified, we can proceed with the next core structure element.

Running Example:
We select the *activity checkPaymentData* as core structure element we want to analyze. It is unclassified at this point of time.

In case the core structure element was not already classified, we now select the according *law structure element hierarchy*. We use the hierarchies we derived from different laws when modeling them using Law Patterns. The person hierarchy corresponds to active and passive stakeholders of a core structure, the activity hierarchy to the activities of a core structure, and the subject hierarchy to the assets of a core structure. For the core structure element at hand we iterate the corresponding hierarchy. From the set of law structure elements of the corresponding hierarchy, which were not already checked, we *select* one of the law structure elements with the highest level[29] as *a possible classifier*. We start at the highest level and follow a bottom up direction towards the root of a hierarchy. There are two reasons for following a bottom up direction. First, we want to classify the core structure element at hand as precise as possible. Second and more important is the fact that from our experience requirements engineers have problems to understand the legal terms which are very generic. In consequence, the discussions between requirements engineers and the legal experts tend to be fruitless when starting with highly generic legal terms, but the discussion significantly improves when starting with the most refined legal terms. As the highest levels of a hierarchy contain the law structure elements refined most, it is reasonable to start with them first. As we iterate over the law structure elements of a hierarchy, it will happen that we select a law structure element which has a refinement which is already added as valid classification. In this case the current law structure element is *already covered*. Hence, we skip the element.

In case we do not skip the *law structure element (classifier)*, we *check* whether the selected law structure element is a valid *classifier* for the core structure element or not. For this check the requirements engineers and legal experts have to collaborate, because the requirements engineers know the meaning of the terms used for core structure elements and can explain them to the legal experts. Vice versa, the legal experts are able to interpret the law terms in the hierarchies and check their applicability to a core structure element. Note that for checking the core structure element at hand, we have to consider the element in its context defined by the complete core structure. It might happen that another core structure element influences the classification of the core structure element at hand. For example, data is only classifiable as personal data if an individual is entitled to this data. In case the *classifier matches* the core structure element, we *add the classification* for the core structure element at hand.

Running Example:
The according law structure element hierarchy for our *activity checkPaymentData* is the law structure element hierarchy for activities as introduced in Section 15.1[30]. The full law structure

[29] We assume that the hierarchies are trees. Then the following definition of level applies: "The level of a node is defined by 1 + the maximum number of connections between the node and the root."

Level	Law Structure Element	Already Covered	Match	Direct Refinements
4	automated retrieval	✖	✖	-
	abroad transfer	✖	✔	-
3	aliasing	✖	✖	
	rendering anonymous	✖	✖	automated retrieval
	transfer through inspection or retrieval	✔	✔	**abroad transfer**
	transfer through transmission	✔	✔	**abroad transfer**
	transfer anonymized	✖	✖	
2	commission collect	✖	✖	
	monitoring	✖	✖	
	commissioned use	✖	✖	
	modification	✖	✖	aliasing, rendering anonymous
	erasure	✖	✖	
	blocking	✖	✖	
	transfer	✔	✔	transfer trough inspection or retrieval, transfer through transmission, **abroad transfer**, transfer anonymized
	storage	✖	✖	
	commissioned processing	✖	✖	
1	collect	✖	✖	commissioned collect, monitoring
	use	✖	✖	commissioned use
	automated decision making	✖	✖	
	process	✔	✔	modification, erasure, blocking, transfer, storage, commissioned processing
0	automated processing	✔	✔	collect, use, automated decision making, process

unmatched law structure element, matched but already covered law structure element, **matched and not covered law structure element**

Table 16.6.: *Classification of the activity checkPaymentData*

element hierarchy for activities according to the BDSG is shown in Fig. E.2[31] in the appendix. The activity classifiers in the highest level are *automated retrieval* and *abroad transfer*. Automated retrieval is not a match, as the BDSG distinguishes between retrieval and transmission and a digital transmission like checkPaymentData is not a retrieval as defined by the BDSG (see Table 15.2[32] for the definition). The **abroad transfer** matches as it includes both activities (retrieval and transmission) as long as borders of Germany are crossed (see definition in Table 15.3[33] and law structure element hierarchy for activities as shown in Fig. 15.4[34]). Note that here the context given by the rest of the core structure is relevant. The fact that the **Gateway Provider** can reside outside of Germany makes *checkPaymentData* classifiable as abroad transfer. We add this **abroad transfer** as classifier for *checkPaymentData*. For the next level of the law structure element hierarchy, we have to check the activity classifiers *aliasing, rendering anonymous, transfer through inspection or retrieval, transfer through transmission, and transfer anonymized*. Aliasing, rendering anonymous, and transfer anonymized are not matching. And transfer through inspection or retrieval, and transfer through transmission are already covered as they are refined by the match **abroad transfer**. The result of all iterations are shown in Table 16.6. Each row contains information about the level of the law structure element at hand, its name, if it is already covered by one of its refinements, if it matches, and its direct refinements.

After we have iterated over the complete hierarchy and no possible *classifier is left*, we continue with the next core structure element of the Law Identification Pattern core instance at hand. In case no core structure *element* is left, we *model the classifications* for the law identification pattern core instance. This way, we get a *full identification pattern instance*.

[30] Page 267
[31] Page 491
[32] Page 269
[33] Page 271
[34] Page 272

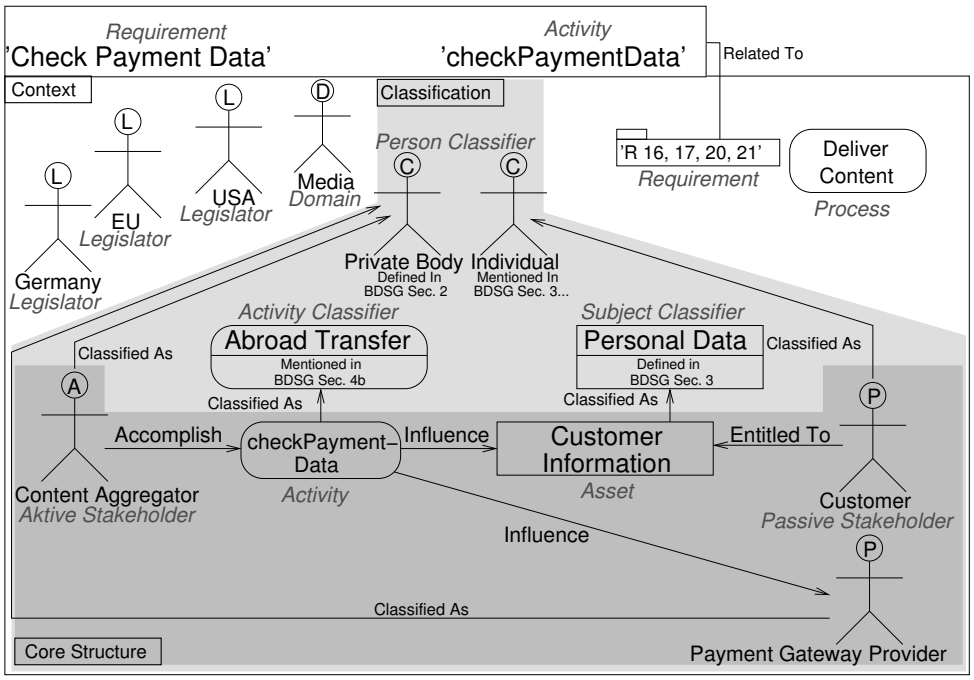


Figure 16.13.: *One Law Identification Pattern Instance (DBC_V2_3) for* `R15`

Running Example:
Figure 16.13 shows the resulting full Law Identification Pattern instance for our example. Beside the *activity checkPaymentData*, which is classified as *abroad transfer*, we classify the *active stakeholder* **Content Aggregator** as *private body*, the *asset* **Customer Data** as *personal data*, the *passive stakeholder* **Customer** as *individual*, and the *passive stakeholder* **Payment Gateway Provider** as *private body*.

16.3. Pattern Matching

After the full instantiation of the Law Identification Pattern for the different requirements, the Law Identification Pattern instances can be matched with the Law Pattern instances. The actual matching is done on the core structure / law structure level. Of course, the matching on the core structure / law structure level is not enough for a relevant match. Also the context has to match. But this has to be decided only initially and once for a law. Whenever there is no match between the legislators and domains given for the law and the system-to-be, there is no need to analyze the law at all.

Running Example:
For the BDSG, the legislator is Germany and the regulated domain is the general public. Germany was already identified as relevant legislator in Chapter 10[35]. The media domain, which was also already identified previously, can be considered as sub-part of the general public. Hence, regarding the context of the BDSG and our system-to-be, the BDSG has to be analyzed.

[35] Page 181

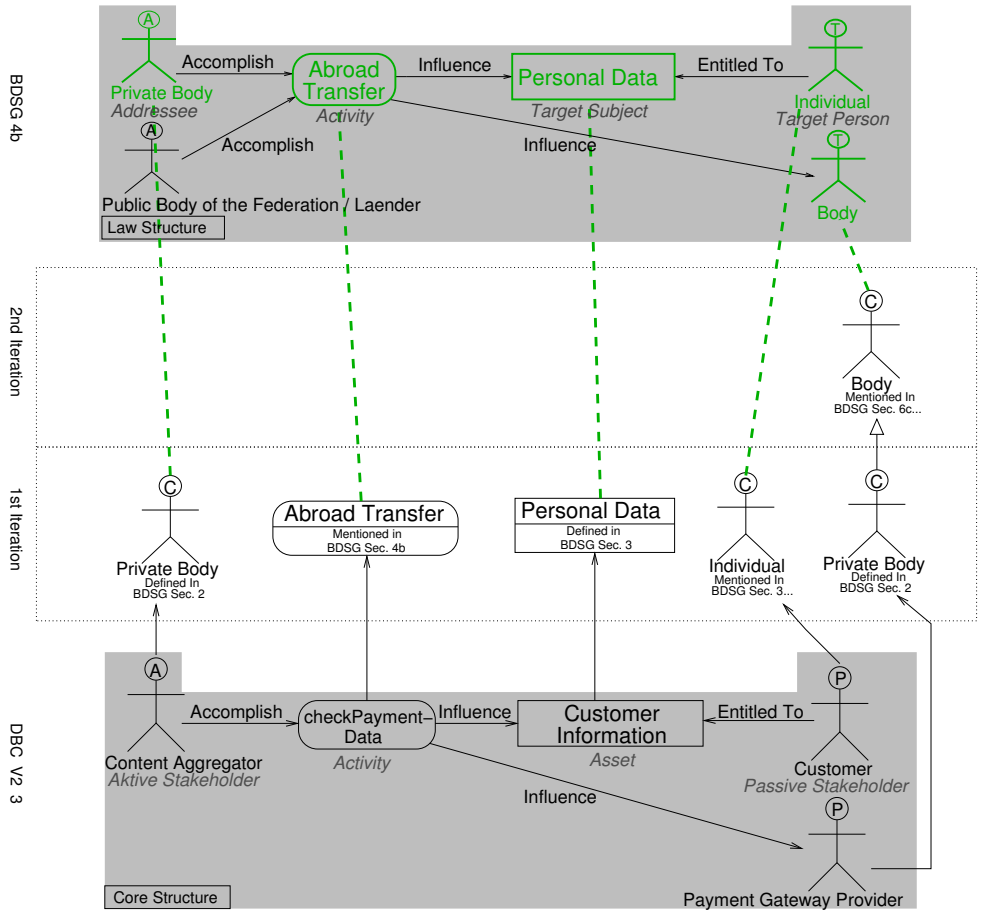


Figure 16.14.: *Matching of Law Identification Pattern Instance (DBC_V2_3) for* `R15` *with BDSG Section 4b*

Next, we have to match the Law Pattern instances for a law with the Law Identification Pattern instance for the system-to-be on the law structure / core structure level. A detailed pseudo algorithm for matching Law Pattern instances which contain the law structures and the Law Identification Pattern instances which contain the core structures is given in Algorithm 1. In the following, we explain the basic idea behind the algorithm. Basically, a Law Identification Pattern instance matches a Law Pattern instance whenever each addressee, activity, target subject, entitled target person, and influenced target person of the facts of the case of a Law Pattern instance has a matching counterpart in the core structure of the Law Identification Pattern instance at hand. For the matching of elements, the elements of the core structure, which are expressed using the terminology of the system-to-be, are replaced by the corresponding terms of the legal domain. The relations between the terms has been established in the previous step. We have a match between two elements whenever they are the same.

Running Example:
For example, abroad transfer in the Law Pattern Instance and checkPaymentData in the Law Identification Pattern, which is classified as abroad transfer (see Figure 16.14)

Or we have a match whenever the classifiers of the core structure element can be generalized to match the law structure element at hand.

Running Example:
For example, the payment gateway provider, which is classified as private body, can be generalized to a body to match the element body within the law structure of the BDSG 4b Law Pattern instance (see Figure 16.14).

Note that for a match all element groups (addressee(s), activity(ies), target subject(s), entitled target person(s), and influenced target person(s)) of the law structure have to be covered, but not every element.

Running Example:
For example, the BDSG 4b defines two addresses, private body and public body, but for a match only one of them has to be covered for a valid match (see Figure 16.14).

Additionally, a law structure might not have elements for all element groups, because target subject, entitled target persons, and influenced target persons are optional. If an element group is empty, it is considered as matched by default. The other way round, a core structure can contain more elements than needed to match a law structure. Hence, we are not looking for a complete match in both directions, only the full coverage of the law structure is of relevance.

Running Example:
Figure 16.14 shows a full match between the BDSG Section 4b Law Pattern Instance and a core structure generated for the requirement `R15`. In the first iteration of Algorithm 1 for the Law Pattern instance, the element groups addressees (private body), activity (abroad transfer), target subject (personal data), and entitled target person (individual) are covered. For the influenced target person, we have to generalize the classifier private body only once to get a match. In the end, we have a full match here.

Algorithm 1: *Matching of Law Identification Pattern instances with Law Pattern Instances*

Require: Law Pattern instances (Set(LawPattern) $insLIPs$)
Require: Law Identification Pattern instances (Set(LawIdentificationPattern) $insLPs$)
Require: Activity Classifier Hierarchy (ActivityHierarchy $hieAC$)
Require: Person Classifier Hierarchy (PersonHierarchy $hiePC$)
Require: Subject Classifier Hierarchy (SubjectHierarchy $hieSC$)
Ensure: Returns a set of pairs of Law Identification Pattern instances and Law Pattern instances which matched (Set(Pair(LawPattern, LawIdentificationPattern)) $foundMatches$)

```
1:  Set(Pair(LawPattern, LawIdentificationPattern)) foundMatches = ∅
2:  for all LawPattern insLP ∈ insLPs do
3:      Set(Person) insLPAdd = insLP.getAddressees()
4:      Set(Activity) insLPAct = insLP.getActivities()
5:      Set(Subject) insLPTarSub = insLP.getTargetSubjects()
6:      Set(Person) insLPTarPerEnt = insLP.getTargetPersonsEntitledTo()
7:      Set(Person) insLPTarPerInf = insLP.getTargetPersonsInfluence()
8:      for all LawIdentificationPattern insLIP ∈ insLIPs do
9:          Boolean matchAdd = FALSE
10:         Boolean matchAct = FALSE
```

```
11:         Boolean matchTarSub = FALSE
12:         Boolean matchTarPerEnt = FALSE
13:         Boolean matchTarPerInf = FALSE
14:         if insLPTarSub == ∅ then
15:             matchTarSub = TRUE
16:         end if
17:         if insLPTarPerEnt == ∅ then
18:             matchTarPerEnt = TRUE
19:         end if
20:         if insLPTarPerInf == ∅ then
21:             matchTarPerInf = TRUE
22:         end if
23:         Set(Person) insLIPActSta = insLIP.getActiveStakeholdersClassifier()
24:         Set(Activity) insLIPAct = insLIP.getActivitiesClassifier()
25:         Set(Subject) insLIPAss = insLIP.getAssetsClassifier()
26:         Set(Person) insLIPPasStaEnt = insLIP.getPassiveStakeholdersClassifierEntitledTo()
27:         Set(Person) insLIPPasStaInf = insLIP.getPassiveStakeholdersClassifierInfluence()
28:         changed = TRUE
29:         repeat
30:             if insLPAdd ∩ insLIPActSta != ∅ then
31:                 matchAdd = TRUE
32:             end if
33:             if insLPAct ∩ insLIPAct != ∅ then
34:                 matchAct = TRUE
35:             end if
36:             if insLPTarSub ∩ insLIPAss != ∅ then
37:                 matchTarSub = TRUE
38:             end if
39:             if insLPTarPerEnt ∩ insLIPPasStaEnt != ∅ then
40:                 matchTarPerEnt = TRUE
41:             end if
42:             if insLPTarPerInf ∩ insLIPPasStaInf != ∅ then
43:                 matchTarPerInf = TRUE
44:             end if
45:             insLIPActSta = hiePC.getParents(insLIPActSta)
46:             insLIPAct = hieAC.getParents(insLIPAct)
47:             insLIPAss = hieSC.getParents(insLIPAss)
48:             insLIPPasStaEnt = hiePC.getParents(insLIPPasStaEnt)
49:             insLIPPasStaInf = hiePC.getParents(insLIPPasStaInf)
50:             if insLIPActSta == ∅ AND insLIPAct == ∅ AND insLIPAss == ∅ AND
    insLIPPasStaEnt == ∅ AND insLIPPasStaInf == ∅ then
51:                 changed = FALSE
52:             end if
53:         until (matchAdd AND matchAct AND matchTarSub AND matchTarPerEnt
    AND matchTarPerInf) OR (NOT changed)
54:         if matchAdd AND matchAct AND matchTarSub AND matchTarPerEnt AND
    matchTarPerInf then
55:             foundMatches.add({insLP, insLIP})
56:         end if
57:     end for
```

58: **end for**
59: **return** $foundMatches$

16.4. Legal Revision

In this step, the actually relevant dictates of justice are identified by legal experts. The automatic matching as described in the previous section only finds potentially relevant matches between legal dictates of justice and requirements. The reason is, that the matching is not able to consider the full context of the dictate of justice. The context of a dictate of justice and the law it is contained in is very broad, starting from the cross references within a law, over references between laws, to actually judged cases based on this particular dictate of justice and the ongoing discussion in the legal community and literature. All this information needs to be known, connected, and interpreted to judge whether a dictate of justice is really relevant for a requirement. This is a task only a legal expert can conduct.

What we have achieved so far by applying our law identification process, is to prepare this step and to ease the work of the legal experts. The terms used for expressing the requirements are already mapped to the corresponding legal terms. Hence, the legal experts can comprehend the legal implications of a requirement easily. Furthermore, the requirements are already linked to specific laws and dictates of justice. In consequence, the legal experts can focus on the already identified relations and do not need to establish them by themselves. Additionally, the Law Identification Pattern instances already show the cross references between dictates of justice within one law or other laws. Hence, the legal experts can quickly get an overview of this part of the context of the dictate of justice at hand. As a result, a much lower effort has to be taken by the legal experts to conduct the step of legal revision.

The automatic matching as described in the previous section is designed in a way that it has a perfect recall but might lack some precision. The validation done so far indicates that the design was correct as the recall in an experimental simulation was 100% while the precision was 94% (for a detailed discussion see Chapter 17[36]). With respect to this observation, the step of legal revision is about removing the false positives. Hence, if false positives are acceptable, this step might be skipped.

16.5. Adjustment of Requirements

After finishing the legal revision, we have a set of requirements which impose the consideration of certain dictates of justice and the according laws. Each of the laws, dictates, and requirements has now to be discussed by the requirements engineers and the legal experts. The three main activities within this discussion are *decide on consideration*, *decide on measure*, and *modification of requirements*.

The first activity *decide on consideration* is about deliberating whether the risk of being non-compliant to a certain dictate of justice is acceptable considering the effort to be taken for achieving compliance. Here, we need the expertise of the requirements engineers as well as the legal experts. On the one hand, the legal experts bring in the knowledge about the penalties to be expected for being non-compliant, and the probability of being sued successfully as well as knowledge about the actions to be taken to be compliant. On the other hand, the requirements engineers bring in the knowledge about the impact of these actions to be taken for being compliant from a system perspective. The relations between dictates of justice and the requirements of the system-to-be help both sides to comprehend the implications for the system-to-be. Note that in this step the discussion is on a rather abstract level and that an

[36] Page 307

action to be taken does not necessarily mean to change the system-to-be (other measures might be taken on organizational, contractual or assurance level). Hence, it is not the aim to already discuss specific requirements, but to remove all those dictates of justice for which the expected risk is already acceptable without taking action. We will not go into detail on this particular step, as there are already well established solutions for such a discussion such as Legal CORAS (Lund, Solhaug, and Stolen, 2011 [245]; Mahler, 2010 [249]). The result of the activity "decide on consideration" is a reduced set of dictates of justice, which contains all the dictates of justice which were identified as relevant and for which a measure has to be taken.

Running Example:
For our example match `R15` and BDSG Section 4b, the probability of being successfully sued is regarded as high by legal experts. For example, the general guidelines for using cloud services proposed by data protection commissioners of several German states specifically mentions BDSG Section 4b as one dictate of justice which, in case it is harmed, will lead to successful sues (Budszus, Berthold, Filip, Polenz, Probst, and Thiermann, 2014 [82]). Also other sources discuss BDSG Section 4b as being important in data protection cases (Plath, 2014 [314]). The penalties for being non-compliant are severe. A fine for a single breach of law will at least be 50.000 €, but has to be higher than the assumed economic benefit gained by being non-compliant (BDSG Section 43 (Bundestag der Bundesrepublik Deutschland (Lower House of German Parliament), 2009 [88])). Additionally, the illegal activities must be stopped immediately, which often means that parts of an IT system have to be turned down. In case it can be proved that one or more natural persons were aware of being non-compliant and in the position to stop the illegal practice, these persons can be personally punished in terms of further fines or even imprisonment up to 2 years. Hence, BDSG Section 4b has to be considered as the risk of being non-compliant is too high.

Next, we have to *decide on the measure* to be taken. Here, several kinds of measures are available. On an organizational level, measures can be to change processes or organizational structures. For example, for being compliant with some sections of the BDSG it is enough to establish a data protection officer and processes which involve the data protection officer. On a contractual level, some obligations can be moved to third parties. For example, in an outsourcing scenario, some obligations can be moved to the outsourcing partner such as the correct storage of data. The obligations moved must be made explicit in a contract. Which obligations can be moved and which bodies are regarded as acceptable contract partners differs from law to law. A third kind of measure is to have insurance. Hence, as long as the consequence of being non-compliant is only a monetary one, it is often possible to have an insurance which pays the costs of a sue. A fourth kind of measure is on the IT landscape level. For example, to replace a component / system by another component / system or move a data-center to another location. The last measure is to actually change the requirements.

Running Example:
In our case, there is no possibility to change a process or the organization to be compliant. The payment data has to be checked, and there is no way to establish a new department which replaces the payment gateway providers. On the contractual level, it is not possible to move this obligation implied by BDSG Section 4b to the payment gateway providers. All of them also operate, for example, in the US which makes them fall under US law. Currently, the US cannot be considered as country with a data protection level which is the same or higher than the one established by the BDSG (Budszus et al., 2014 [82]). But this would be necessary for moving

the obligations implied by BDSG Section 4b to a third party (Bundestag der Bundesrepublik Deutschland (Lower House of German Parliament), 2009 [88]; Plath, 2014 [314]). There is also no way to insure against sues based on the BDSG. Changing the IT landscape is also not possible, as the service provided by the payment gateway providers cannot be replaced by another service or internal system. Hence, we have to change the requirements.

For the *modification of requirements*, we also have several options. First of all, by removing the requirement, because this also removes the dictates of justice it imposes.

Running Example:
To remove the requirement is not an option, because without payment the whole business model of the content aggregator does not work.

A second option is to actually modify the requirement. Such a modification aims at those elements which are relevant for a match with a dictate of justice. For example, reducing the data necessary for fulfilling a requirement can remove a match, because the asset of the corresponding Law identification Pattern instances changes and might not match the target subject anymore. The same might be achieved by modifying the active stakeholder or the activity. Another way of modifying a requirement is to add aspects which are required by the dictate of justice at hand for being compliant.

Running Example:
In our case, we could try to remove all the personal data from the payment data sent to the payment gateway. This would remove the match with the BDSG Section 4b, because the target subject of the Law Identification Pattern instance would not be matched any longer (see Figure 16.14). Unfortunately, the payment gateway will not be able to operate properly without the personal information. Also changing the activity or the active stakeholder is not possible. But the transmission of the personal data about an individual is allowed in case of an explicit

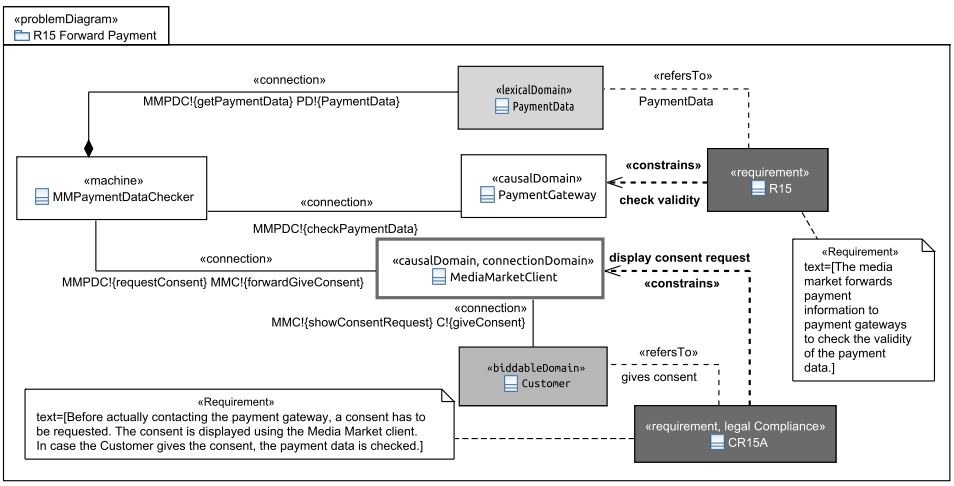


Figure 16.15.: *Problem Diagram for `R15` after Modification*

consent by this individual. This consent has to include the reason for transmission and the consequences of allowing or disallowing the transmission. Hence, we have to modify `R15` in a way that it requires a consent of the customer. Figure 16.15 shows the resulting problem diagram.

Sometimes it is also possible to add further requirements to be compliant.

Running Example:
For being compliant with other sections of the BDSG, there must be a way to delete personal data on request of the individual which is entitled to this personal data. This clearly introduces new requirements, because such a deletion process was not planned for the system-to-be.

16.6. Conclusion

In this chapter we discussed the steps which have to be conducted for each system-to-be to identify relevant laws and adjust the requirements of the system-to-be accordingly. For the first step "instantiation of Law Identification Pattern (Core+Context)", we introduced a structured method for transforming functional requirements into Law Identification Pattern instances. The transformation makes use of problem diagrams for structuring the functional requirements, problem frames for transformation instructions, domain knowledge for considering the context of the system, and questionnaires for refining the domain knowledge. Next, we explained how to close the gap between the legal wording of laws and the technical terms used for expressing the requirements of the system-to-be. For the actual pattern matching, we proposed and discussed an algorithm. The result of the matching has to undergo a legal revision, as the matching based on the modeled information is not able to reflect all the information for judging the relevance of a particular dictate of justice for a requirement. Last, we explained how to check the remaining matches between requirements and laws if they should be considered, which measures can be taken to be compliant, and how to actually change the requirements.

The contributions of this chapter are:

- Reuse of results of an existing requirements engineering (here problem frames) approach for law identification.
- Transformation cards, which enable software engineers to
  - identify the problem class of the requirement at hand.
  - identify the needed domain knowledge for the transformation of problem diagrams to Law Identification Pattern instances.
  - obtain instructions how to model the domain knowledge.
  - execute the transformation.
- A tool for assisting the requirements engineer in using the transformation cards.
- An algorithm for the matching of requirements and laws.
- A preparation of requirements and matching dictates of justice which eases the legal revision.
- Guidance for deciding which requirements to change and how to change them.

Chapter 17

Application of and Reflections on Legal Compliance Requirements Elicitation

In this chapter, we take a look at the sufficiency of the solution proposed in Chapter 14[1], Chapter 15[2], and Chapter 16[3] for the identification of laws and the subsequent adjustment of requirements. In Section 17.1, we present a validation in terms of experimental simulations, which investigate the recall and precision of our method regarding the laws identified. The case used is the voting system as introduced in Section 5.2.1[4]. The experiment and according results were already presented in Faßbender and Heisel (2013 [133]), and Faßbender and Heisel (2014 [134]). Finally, we critically discuss our presented work considering the important aspects for a solution in legal compliance requirements engineering as identified in Section 14.3.2[5], and outline future work in Section 17.2[6]. Section 17.3[7] concludes this chapter.

17.1. Validation

For the validation of our proposed method we analyzed the voting system as introduced in Section 5.2.1[8] in an experimental simulation. For the purpose of the experiment we considered all requirements related to the voter. We applied the process as described in Section 14.5[9] and integrated the transformation cards into the process. We matched the resulting Law Identification Pattern instances with selected laws of Germany. These laws were modeled in terms of Law Pattern instances for the voting system case or were already modeled for other purposes. There were four hypotheses to be investigated, namely *"H1: The transformation cards are sufficient to be integrated in the overall identification process as described in Section 14.5[10]."*, *"H2: The effort to be taken for conducting the identification process along with the transformation cards is reasonable."*, and *"H3: Using the complete identification process as described in Section 14.5[11] along with transformation cards leads to an identification of all relevant laws."*. When we were conducting the first run of the experimental simulation, we had no tool support available. But we

[1] Page 239
[2] Page 267
[3] Page 281
[4] Page 77
[5] Page 246
[6] Page 314
[7] Page 317
[8] Page 77
[9] Page 262
[10] Page 262
[11] Page 262

observed that tool support might improve some results which were not in favor of our proposed method. Hence, we made a re-run for the voting system using a newly developed tool. The re-run should ensure that the integration of the tools does not negatively influence the results obtained when applying our method, while mitigating some of the flaws observed in the first run. Hence, we checked the hypothesis *"H4: Using tool-support improves the execution of the overall identification process as described in Section 14.5[12] and the usage of the transformation cards as introduced in Section 15.2[13]"*

To decide the first, the second and the fourth hypotheses, we tested the usage of transformation cards by integrating the transformation process in the overall identification process as described in Section 14.5[14]. To be able to discuss the sufficiency of and the effort spent for the process, we tracked the generation of the different outputs in terms of number and time spent in both runs of the experimental simulation. For both runs we also documented our experiences while conducting the method. The result is more of a qualitative than a quantitative nature.

To answer the third hypothesis, we conducted a literature research about the voting system and the relevant German laws for this matter. The main source was the judgment of 2009 by the Federal Constitutional Court of Germany (2009 [140]), followed by discussions with several domain and / or legal experts. These insights led to expectations whether a requirement will match with a particular law or not. These expectations were documented in terms of a table listing the expectations for each law and requirement. This table was compared to the matching produced by our proposed method. The result is of quantitative nature in terms of false positives and false negatives, and evidence by the number of matches.

For the validation, we excluded some laws even though they were identified as relevant based on the literature research and discussions. We selected only the highly relevant laws as discussed by the Federal Constitutional Court. These laws are necessary to find the weaknesses of our method in terms of false negatives. A false negative would be a missing match with a law for a certain requirement. To identify false positives, we also added laws which are somehow related to voting systems, but not relevant. Hence, here we expected to find matches which are not of real relevance. For the validation, we selected the four following laws:

- The BDSG as highly relevant law concerning personal data.
- The BWahlG (Bundeswahlgesetz), which is the law for federal state elections in Germany and also highly relevant.
- The SigG (Signatur Gesetz), a law which regulates the use of digital signatures. This particular law was selected not due to its relevance, but it is related in terms of the technological background. Therefore, it is interesting whether such a law, which is only related in the used wording, will match or not.
- The PassG (Pass Gesetz), which regulates the use of passports in Germany. This law is clearly irrelevant, even though the passport is a possible authentication means during elections.

Table 17.1 shows the results of using the transformation cards. In the first two columns, the requirement and its name are listed. In the next column the identified problem frame for the requirement at hand is listed. The next column states how many core structures were generated by executing the transformation described by the transformation card for the problem frame. While executing the transformation it turned out that applying the transformation rules generates a noticeable number of duplicated core structures. Hence, we added this information

[12] Page 262
[13] Page 275
[14] Page 262

Requirement	Name	Problem Frame	Core Structures		BDSG		BwahlG		SigG		PassG	
			Generated	Redundant	Matched	Relevant	Matched	Relevant	Matched	Relevant	Matched	Relevant
R1	Identification and authentication of the voter	Query	16	8	8(4)	2(2)	0(0)	0(0)	16(8)	0(0)	0(0)	0(0)
R2A	Show the ballot	Query	20	16	12(2)	4(1)	0(0)	0(0)	0(0)	0(0)	0(0)	0(0)
R2B	Completion of the ballot	Simple Workpiece	12	7	8(3)	2(1)	4(2)	4(2)	0(0)	0(0)	0(0)	0(0)
R2C	Correction of the ballot	Simple Workpiece	12	7	8(3)	2(1)	4(2)	4(2)	0(0)	0(0)	0(0)	0(0)
R3	Initiation of vote casting	Commanded Transformation	20	6	5(3)	5(3)	12(7)	8(4)	0(0)	0(0)	0(0)	0(0)
R4A	Hasty Voting Protection	Query	24	14	16(6)	4(2)	5(4)	4(3)	0(0)	0(0)	0(0)	0(0)
R4B	Cast confirmed vote	Commanded Transformation	8	4	6(2)	1(1)	6(2)	6(2)	0(0)	0(0)	0(0)	0(0)
R4C	Acknowledgment of vote	Model Display	14	5	3(2)	1(1)	0(0)	0(0)	0(0)	0(0)	0(0)	0(0)
R4D	Delete completed ballot	Simple Transformation	12	7	7(3)	2(1)	0(0)	0(0)	0(0)	0(0)	0(0)	0(0)
R5	Abort of the polling process	Commanded Display	6	4	0(0)	0(0)	0(0)	0(0)	0(0)	0(0)	0(0)	0(0)
Sum			**144**	**78**	**73(28)**	**23(13)**	**31(17)**	**26(13)**	**16(8)**	**0(0)**	**0(0)**	**0(0)**

Table 17.1.: *Validation Results: Core Structure Level*

in the next column. Next, the four laws are listed. For each law we tracked how often it was matched by a Law Identification Pattern instance and whether this match was relevant or not. The number before the parentheses states the number considering all Law Identification Pattern instances, which include the duplicates. The number within the parentheses states the number without the duplicates. This table allows some reasoning about the transformation cards and their sufficiency for assisting the instantiation of Law Identification Patterns.

From our observations, the transformation cards integrated well in the overall law identification process. After the core structures were generated, no further modifications were required to fully instantiate the Law Identification Pattern instances. The preparations of the core structures were more structured and detailed and therefore less error-prone compared to the hands on instantiation we used previously.

A major downside we observed when executing the process manually is the sheer number of generated core structures. Generating the core structure takes a significant amount of time and afterward not every core structure is necessary for a successful matching. The high share of duplicates worsens the situation even more, because they devalue more than half of the work (78 duplicates out of 144 as shown in Tab. 17.1). From our experience, none of the non duplicated core structures can be removed as they are all necessary for a detailed detection of relevant laws. Sometimes only the minor differences between core structures revealed the most important parts of a requirement for the relevance of a law. These differences then help to understand how to address the law. Furthermore, we could not identify common characteristics for filtering out the core structures which match all relevant laws. The laws are too different to do so. Hence, the big number of core structures is necessary and even helpful, but the effort to generate and analyze them manually was significant, and executing the generation was regarded as tedious and not acceptable.

The tool support mitigates the most significant flaws mentioned, because the core structures are generated automatically, the additional effort spent is zero at this point. Additionally, the tool is able to remove the duplicates. Furthermore, the matching can be done automatically too, which even more lowers the effort.

Speaking of the effort, there are several aspects to consider (see Table 17.2). Note that in the following paragraphs we discuss the fully manual application of the process. Later on

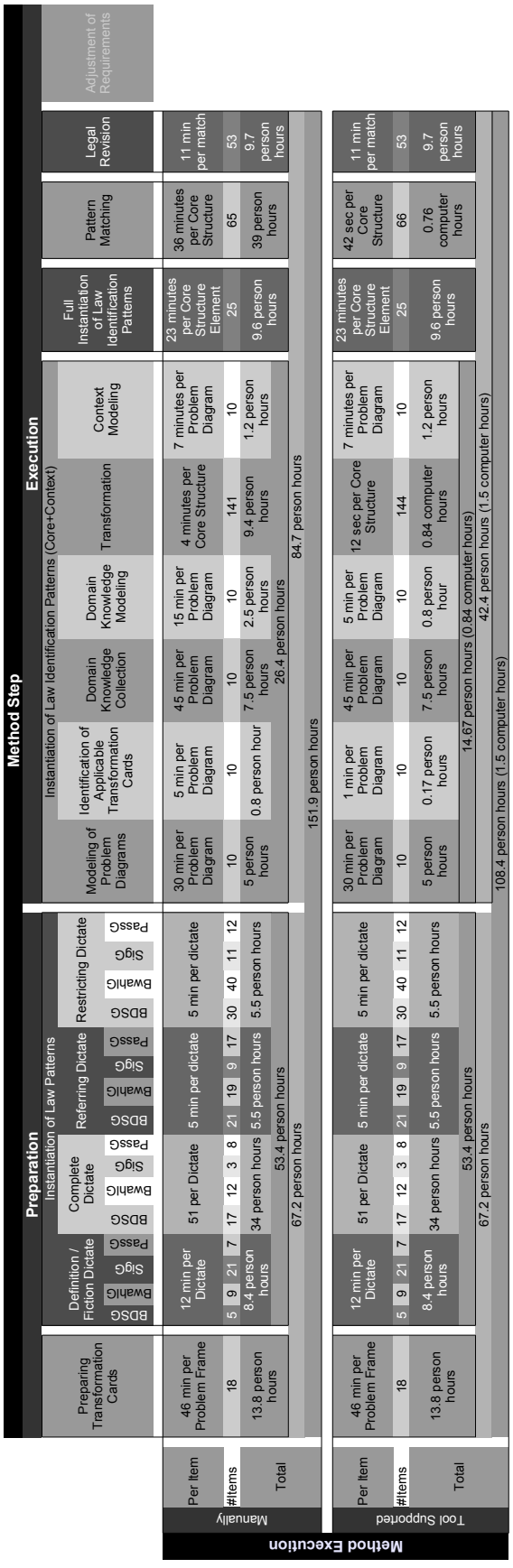


Table 17.2.: *Effort spent for conducting the method*

we will discuss the improvements reached using the tool support. First of all, the preparation of transformation cards and laws consumes a significant amount of time and effort. For preparing the transformation cards for the 18 unique problem frames[15] as enumerated by Côté et al. (2008 [103]), we needed 13.8 hours (41 minutes per problem frame). For modeling a law using the Law Pattern, we observed that the effort to be taken depends on the law at hand as the number of dictates within laws differs to a large extent. Additionally, the effort to be spent also depends on the actual dictate types found in a law. For example, a referring dictate takes only 5 minutes whereas a complete dictate takes 51 minutes. In total, it took 53 hours to model the 4 laws we wanted to consider, which makes a total of 67.2 hours for the preparation steps. This is a noticeable effort, but we have to keep in mind that the results of these steps are re-usable. Under this light, the effort is high but acceptable.

For conducting the steps which are executed for each system-to-be we observed that the matching of problem diagrams with problem frames is straightforward and takes almost no time (5 minutes per problem diagram, which makes it 0.8 person hour in total). The modeling of the problem diagrams by an experienced user took 5 person hours (30 minutes for one problem diagram). This seems reasonable if the law identification gives sufficient results afterward. Answering the questionnaire and modeling the resulting information takes some more time. But this process of answering and modeling speeds up significantly with a growing number of already analyzed requirements, because the questions consider the domains directly. And as the number of domains is limited, so is the information needed about and modeled for them. Thus, most information is already known and modeled for later requirements. It took 7.5 person hours for this step including discussions (about 45 minutes for one problem diagram). This seems to be a significant amount of work, but the information collected is crucial for the success of the application of the transformation card. Hence, we spent some more time for these discussions and searching for the necessary information (e.g. in the protection profile (Volkamer and Vogt, 2008 [383])). Setting up the core structures is an easy task, but also time consuming. It took 9.4 person hours to generate all core structures by hand, which is about 4 minutes per core structure. The modeling of the context only takes 1.2 person hours, because all information needed is already known when following the overall process proposed in this thesis, and the context modeled is not exclusive for one core structure but the same for all core structures belonging to one problem diagram. In total, executing the instantiation of the core and context part of Law Identification Pattern instances took 26.4 person hours, which makes it about 4 person days for this step. The full instantiation of Law Identification Patterns took 9.6 person hours, which makes it 23 minutes for each unique core structure element including the traversal of the according hierarchies and the discussions necessary to find the correct mapping. The manual pattern matching took quite long. Analyzing one core structure and finding the according dictates of justice took 36 minutes. As the number of core structures was high (65), the overall time spent summed up to 39 person hours. The matching itself is quite easy but a boring task, nevertheless one has to be focused and concentrated on the task to avoid mistakes. The legal revision of the found matches (53) was conducted quite fast with 11 minutes per match, which makes only 9.7 hours for all matches. Here, all the collected information and already established mappings helped a lot to speed up the process. Note that for the simulation we did not do a risk analysis as we were only interested in removing the false positives. We also did not conduct the adjustment of requirements as we had no benchmark to test the results against, which means that we have no evidence to judge the sufficiency of this step. Overall, the steps (apart from the adjustment) to be conducted for each system-to-be took 84.7 person

[15] Côté et al. (2008 [103]) enumerate some more, but these additional problem frames are just variants of the unique problem frames. For example, a variant contains two lexical domains instead of one for the unique problem frame, but the two lexical domains can easily be merged to turn the variant into the unique problem frame.

hours, which makes 151.9 person hours for all steps of the identification process.

This effort seems to be significantly high, but there are some things to consider: First, the preparation steps do not have to be conducted each time. Hence, when the necessary inputs generated by the preparation steps are already available, an execution of the process takes only 83.5 hours. Second, modeling problem diagrams enables one to use several other methods available for analyzing problem diagrams. For example, they can be analyzed for interactions (Chapter 18[16]), security (Chapter 13[17]), privacy (Beckers, Faßbender, Heisel, and Meis, 2012 [42]) or standard-related issues (Beckers, Côté, Faßbender, Heisel, and Hofbauer, 2013 [45]). In an ideal case, the problem diagrams are already available and not specifically modeled for the law identification. Third, the effort for the elicitation of law-relevant information has to be invested even if someone uses another method to find relevant laws. The additional knowledge which is not contained in the requirements is crucial from a legal point of view. Thus, the effort related to the use of transformation cards is limited to the modeling of problem diagrams and modeling of the collected information. And this pays off as it enables the guided and structured generation of the core structure, matching and analysis of the matches. Forth, some steps speed up significantly when using the transformation cards. From our experience, the domain knowledge collection is one of the speed up steps. Without the guidance of the transformation cards and the concrete link between requirements and questions, collecting legal domain knowledge is less focused and structured. This results in more iterations and even in missed domain knowledge. The same observation applies for the full instantiation of Law Identification Pattern instances and the legal revision. The transformation cards prepare these steps in a way that they can be conducted faster and that they are less error prone. Hence, the effort to be spent for the overall process and specifically for the transformation cards seems to be reasonable, as long as the results of the identification process show a perfect recall and acceptable precision.

Result H1: From our experience, the transformation cards are sufficient to be integrated in the overall identification process as described in Section 14.5[18].

"Result H2: The effort to be taken for conducting the identification process along with the transformation cards is reasonable, as long as the results of the identification process show a perfect recall and acceptable precision"

For assessing the precision and recall of the law identification using transformation cards, we set up Table 17.3. The different crosses should be read the following way:

- A normal cross indicates an expected and observed match.
- A bold cross indicates an unexpected but correct match.
- A canceled cross indicates an expected but not observed match and the missing cross turned out to be correct.
- An underlined cross indicates an unexpected and irrelevant match.

For the precision the identification turned out to be remarkable. The precision for the voting system and the four selected laws is at 0.94 (true positive / (true positive + false positive) = 16 / (16+1)). Thus, almost every match points out a relevant law.

For the recall the result is perfect. The recall is at 1 (true positive / (true positive+false negative) = 16 / (16 + 0)). Thus, not a single relevant law is missed.

Having a high recall is more important than a high precision in our case. The method should find all relevant laws. The impact of a missed law is much more serious to the development and success of the system-to-be than the extra effort spent on the legal revision for an irrelevant

[16]Page 321
[17]Page 221
[18]Page 262

Law	Requirement									
	R1 identification voter	R2A show ballot	R2B complete ballot	R2C correct ballot	R3 init vote	R4A hasty vote protection	R4B caste Vote	R4C acknowledge vote	R4D clear buffer	R5 abort vote
BDSG	X	**X**	X	**X**	X	**X**	X	**X**	X	
BwahlG		~~X~~	X	~~X~~	X	~~X~~	X	~~X~~		
SigG	X									
PassG										

X = expected, **X=unexpected but correct**, ~~X=expected but not correct~~, X=unexpected and incorrect

Table 17.3.: *Validation Results: Expected and found matches*

law. Hence, having recall value near to 1.0 is the main objective of our method. But a precision of 0.94 adds some surplus value. In terms of precision and recall, it is reasonable to use our method.

Compared to our expectations based on the requirements and legal insights alone, using the identification method is superior. For the BDSG we neglected the fact that the information about the candidates and their relation to parties is personal information which falls under the BDSG. In fact, this issue is not of high importance as this information is publicly available. But nevertheless, it makes the BDSG relevant for these requirements. For the BWahlG it turned out that this law and its sections only deal with the expression of opinion alone. Thus, those requirements that are not directly related to the voting itself are not in the focus of this particular law. The only point in favor of our manual prediction is that we rejected the SigG, while it was matched once by the law identification process. Integrating the tool into the method did not change any matching. To test whether there is a statistical difference between the outcomes of our discussion-based matching and the result of our method, we applied a two-tailed paired t-test at the level of 0.05. The null hypothesis to be tested was that there is no difference in the outcome. Unfortunately, we could not reject the null hypothesis. This might be due to the small sample size, because at first sight there is a significant difference. Hence, there is no statistical significant difference, but from our point of view there is a strong indication that our method outperformed the discussion based prediction (0.94 vs. 0.67 for precision and 1.0 vs. 0.67 for recall). All in all, we can say that for the result of the method there was not a statistical significant difference from the discussion-based method observed, and that we have a strong indication that precision and recall of the law identification process is higher than for our discussion-based results.

"Result H3: To sum up, using the identification process along with the transformation cards leads to an identification of all relevant laws with a high precision. Additionally, there is strong indication that it improves the situation compared to an unguided method."

When deciding the fourth hypothesis, we have to consider two different aspects. First, did the tool have an impact on the results of the method itself? Second, did it improve the effort to be spent? For the impact aspect we experienced no significant change. Only in the generated core structures we discovered three additional core structures. They were generated by the tool but overlooked when doing the generation manually. In our case, this did not change anything regarding the matching. Thus, using the tool does not add a benefit compared to conducting the method manually in an accurate way. But we assume that such a benefit would be visible if the method had been executed by inexperienced users or users in a rush. For the effort aspect, Table 17.2 shows a significant improvement. For the steps taken for each system-to-be, we lowered the effort by more than 50%. For the manual method execution we spent 83.5 person hours, while using the tool we only spent 41.2 hours.

Result H4: Hence, the tool improves the execution of the law identification process and transformation cards.

17.2. Discussion

In Section 14.3.2[19] we found, discussed and enumerated some important aspects a solution for legal compliance requirements engineering has to cover. In the following, we will discuss our method under the light of these aspects:

Collaboration Our method explicitly requires and describes the collaboration between requirements engineers and legal experts. The method as described in Section 14.5[20] contains steps which can be conducted by requirements engineers or legal experts alone, while others have to be conducted in collaboration. For those steps, the collaboration is described and guided by the method. Overall, all steps of the method aim at easing the collaboration.

Ambiguity Law Patterns capture the laws and the dictates they contain as they are. There is no necessity to remove ambiguity or to refine dictates to be more precise. In turn, the method presented is designed to have a perfect recall and a fair precision. Hence, false positives due to ambiguity are not seen as problem but anticipated and treated within the method. The interpretation and treating of ambiguity is done at two points. First, when the terms used for the system-to-be are mapped to the legal terms of laws. Second, within the legal revision. In both steps, legal experts are involved, who are able to take into consideration the broader legal context and to interpret the ambiguity under the light of this context and the system-to-be. For both points, the method is designed to ease the work of the legal experts in means of preparing the legal terms as well as the important terms used for the system-to-be, and by already collecting and presenting important domain knowledge. But the final judgment how to resolve an ambiguity is up to the legal experts, which, from our point of view, is unavoidable as in a legal case also legal experts (for example, the judge) do the reasoning.

Bridge the Gap Our patterns for law identification take into account how legal experts and requirements engineers are used to work. They do not require the legal experts or requirements engineers to change the way they are working, nor require them to understand in detail how the opposite side is working. The requirements engineers are working on requirements as they are used to, as well as the legal experts conduct the analysis as they are used to. To bridge the gap, both are only asked to restructure their results in a way using the Law (Identification) Pattern, so that the opposite side can understand and use them.

Wording When instantiating the Law Pattern as well as the Law Identification Pattern, a terminology containing the important legal terms respective the important terms for the system-to-be is built. Hence, the wording is made explicit and is a core part to be considered when applying our method. Moreover, there is an explicit step for connecting the legal wording with the wording of the system-to-be. For creating the mapping, requirements engineers and legal experts have to collaborate to lower the odds of misinterpretation or incompleteness.

Relations The relations between dictates of justice within a law, and between dictates of justice of different laws are considered explicitly when instantiating the Law Pattern. The interpretation of such relations is up to the legal experts. Further relations are currently not considered as we do not provide any means to analyze, for example, judged cases or legal literature. Hence, the most important relations regarding a statute law are captured, because the statutes are the central concern for judging a case, but the broader legal context might need some further research to ease the work of the legal experts even more.

[19] Page 246
[20] Page 262

Traceability The traceability between laws, dictates of justice they contain, and the requirements of the system-to-be is explicitly established. Also the impact of a law is directly visible as the requirements of the system-to-be before the legal compliance analysis are known as well as the modified requirements. To some extent these links and the before- and after-state allow a reasoning what has been done to achieve compliance. But regarding this particular part, the support could be even stronger. Currently, the tracelinks are not modeled explicitly as the law models are separated from the requirements models. The matching only enumerates the matches textually, but does not create the tracelinks modelwise. Another point for improvement is the documentation of the reasoning behind certain decisions. For example, why terms were mapped, why a match was not considered to be relevant, or why a requirement was changed in a certain way, all those decisions are not required to be made explicit when using our method. As this is important for making a stand in a legal case, this indeed is important future work.

Dynamics of laws From our experience, the Law Pattern Instances are rather robust against changes in a law, as they only capture the central elements. As long as such elements are not touched by a change, the change has no impact on the instances. But even in case a dictate of justice changes significantly, a new dictate of justice shows up, or a dictate of justice is removed, the effort for addressing this change is quite low, because only local changes are necessary. There is no need to model the complete law again, but it is sufficient to only look at the change texts and analyze them. On the other side, changes in requirements can be addressed almost automatically when using the transformation card approach. In most cases, the core structures can be generated automatically without any further interaction with the requirements engineer, and the mapping between the terms of the system-to-be and the legal terms is already known. Only in case new domains are introduced or a new problem frame never considered before is identified, additional domain knowledge has to be collected. In consequence, there might occur new terms for the system-to-be which have to be mapped to legal terms. In both cases, when a law changes or the requirements change, it is easy to identify the matches which are new due to the changes. Only these changes have to undergo the legal revision and the adjustment of requirements. Hence, the effort for an iteration of the method is considerably lower than for the initial run of the method.

Tool support Our tooling supports the modeling of laws, the collection of legal domain knowledge and the transformation of requirements as long as problem diagrams are used, and the matching. In case problem diagrams are used, there is also tool support for modeling the requirements. The tool support currently lacks some integration between the different steps. For each step, there is at least a proof of concept, but a holistic tool which integrates all the steps under one umbrella is still missing. Nevertheless, the most crucial parts which need tool support identified for our method are tool supported right now. Namely, the modeling of domain knowledge, the generation of core structures, and the matching. Indeed, there is potential for further research. For example, one can think of a recommender-system which supports the step of mapping terms of the system-to-be to legal terms. Here, one could think of a system which learns, for example, from previous runs for other systems-to-be.

Applicability Our method is suitable for statute-based legal systems and outcome-based laws. In particular, it is suitable for legal systems which follow the idea of the Roman jurisdiction. For most European countries, this is the case. Nevertheless, one should keep in mind that the method was tested using German law only. If it is applicable to laws of other countries, has to be shown in further investigations. But the author is quite confident that

> the method is applicable for further countries, as the legal foundation and the methods used by legal experts are similar to a high degree as long as the legal system is statute-based and the laws only describe the outcome. Additionally, the Law Pattern is not prone to problems caused by difference in language as it only extracts important terms and is not sensitive to formulations, grammar and alike.

Beside the general discussion of the method, we can also take a look at the different steps and discuss whether they are already sufficient for their purpose, and if they can be improved in future work. For the preparation of the law identification there are two steps to consider. First, the creation of transformation cards, and second, the instantiation of the Law Pattern for different laws.

Regarding the transformation cards, we already provide transformation cards for all basic problem frames as described by Jackson (2001 [200]), and the extended list by Côté et al. (2008 [103]). This should be sufficient for all requirements one might face. Even problem diagrams which are more complex (the covered problem frames contain up to 3 different domains) will match the existing transformation cards, as complex problem frames are in the end compositions of less complex problem frames. Nevertheless, more complex problem frames might indicate additional domain knowledge. From our experience, the majority of domain knowledge collected for a complex problem frame is also collected for the frames it is composed of, but in rare cases the relations between the domains of a complex frame raise additional questions. But we also observed that such additional questions were especially found when increasing the complexity from one domain in a problem frame to two domains. From two domains to three domains, we hardly found any new question. Hence, it might be worthwhile to take a look at some problem frames containing four domains and whether additional questions arise or not, because if no new questions arise, the existing catalog can be seen as complete regarding its purpose. Another question regarding the transformation cards is if the questions for collecting the legal domain knowledge and the abilities provided by the tool to model them is complete. Currently, we can say that for the cases we investigated all necessary questions were raised and that we were able to model the according answers. We assume that this observation can be generalized, but only further case studies can provide the needed insights.

For the instantiation of Law Patterns, we provide only four laws. Hence, a much bigger database of Law Pattern instances is needed to analyze a system-to-be for full legal compliance. We already have the tool support in place to make the modeling as convenient as possible. For example, we observed that many legal experts prefer textual modeling over graphical modeling. Hence, the tool supports both ways of modeling. But still, there are improvements possible regarding the modeling itself. The collaboration and sharing support for modeled laws is weak. Currently, a Law Pattern instance model is stored as local xml file. Hence, working at the file at the same time, keeping track of changes, and having a community maintaining a database of laws is not supported directly. But enabling such collaborative work is a crucial factor for the usage of the method within bigger companies or an open community maintaining the tool. Another research direction might be natural language processing to speed up the modeling of laws. It might be possible to extract important terms and even some parts of a core structure automatically. There are some promising works on this topic such as the one by Zeni (2015 [401]). In our opinion, a revision by a legal expert is still necessary, but a preprocessed text might ease the work. Hence, the Law Pattern and the process to instantiate them are already sufficient for their purpose, but there is potential to lower the effort.

For the steps conducted for each system-to-be we have to discuss the instantiation of the Law Identification Pattern (core+context), the full instantiation, the matching, the legal revision, and the adjustment of requirements.

From our experience, the guidance for the instantiation of the Law Identification Patterns (core+context) is already sufficient and mature. The tool is still only a proof of concept and in

its infancy, but overall conducting this step is straightforward and the pain points of modeling the domain knowledge, the actual transformation, and the matching are mitigated by the tool. But this applies only when using problem diagrams, the transformation cards, and all the other inputs collected while conducting context elicitation, process analysis, and so forth, as proposed in this work. When using the more general process as described for the Law Identification Pattern in the Context Pattern catalog (Chapter 8[21]), the guidance provided is less structured and there is no tool support beside the modeling of the Law Identification Pattern instances themselves. Hence, when the input to this step are problem diagrams and one is following the overall process described within the complete thesis, the execution of the instantiation of the Law Identification Patterns is guided and supported pretty well, but for other processes, notations, and inputs there is further research needed.

The full instantiation of the Law Identification Pattern instances is also described and guided well, and we are not aware of any flaws indicating a pressing need for future work. The only point for future work might be a recommender system for the mapping of terms. Maybe it is possible to learn from executions for other systems-to-be which technical terms map to which legal terms on a regular basis.

The tool-based pattern matching is fully convenient and sufficient from our experience and there is no future work.

For the legal revision, the guidance provided by our method is sparse, because for this step the legal experts apply the methods from the legal domain they are used to. Hence, to provide more guidance might not be possible, and even more not reasonable as we do not want to change the way the legal experts work. But still, there is some future work regarding the inputs which ease the work of the legal experts. Currently, we are only modeling and relating requirements and laws. But the legal context given by a law is broader than the law itself. The judged cases considering the law, the discussions about the law in legal literature, and so forth, are also part of the legal context defined by a law. Currently, we are not capturing these parts of the legal context, but it might be worthwhile to model also this legal context, because providing this information explicitly might also ease the work of the legal experts. Moreover, it might be possible to not only identify relevant laws, but also directly identify relevant cases. Overall, there are extensions for this step one can think of, but there are no pressing flaws which must be improved.

The guidance provided for the adjustment of requirements is on a high level. Here, some more guidance might be possible by integrating already existing solutions, or by creating new solutions. For example, we already mentioned that for the risk analysis some solutions already exist (for example, Legal CORAS (Lund et al., 2011 [245]; Mahler, 2010 [249])). There are also some solutions for interpreting laws to derive new requirements or align existing requirements for a specific system-to-be to be compliant (for example, Ishikawa et al. (2009 [196]), and Massey et al. (2010 [256])). Also solutions which enable to assess to which degree a system-to-be is actually compliant (for example, Massey et al. (2009 [255])) might provide some surplus when integrated. To which extent those solutions can be integrated to provide more guidance might be a worthy subject for future investigations. Hence, the current description of the step already enables one to adjust the requirements. Nevertheless it is possible to provide much more guidance for specific sub-steps, which should be topic of future research as well as some more validation specifically for this step.

[21] Page 131

17.3. Conclusion

Overall, we can conclude that our proposed method for law identification serves its purpose. Nevertheless there is still future work. The experimental simulations have shown that the recall of the matching is perfect, while the precision is still high. There are some improvements of the method possible, but the method in its current shape is already usable and producing good results.

The contributions of this chapter are

- the results of two experimental simulations which show
  - that the effort to be spent for using the Law (Identification) Pattern and executing the law identification method as presented in Chapter 14[22] is reasonable,
  - the method identifies relevant laws for the requirements of a system-to-be with a perfect recall and a high precision,
  - and that the tool support developed improves the execution of the method,
- a discussion whether the method presented takes into consideration the important aspects for a legal compliance requirements engineering method as identified in Section 14.3.2[23],
- and a discussion of each step of the method whether there is potential for further improvements and future work.

[22] Page 239
[23] Page 246

Part IV.

Reconciliation

Chapter 18

Interactions And Alternatives

Up to now, we have finished the two phases *understanding the purpose* and *understanding the problem*. Hence, we are entering the *reconciliation* phase. But first, we should take a look at the information collected so far (Section 18.1), before we actually try to detect the interactions between requirements (Section 18.2), and generate alternatives for the conflicting ones (Section 18.3[1]). The latter two sections are based on Alebrahim, Choppy, Faßbender, and Heisel (2014 [6])[2]. Last, we conclude this chapter (Section 18.4[3].)

18.1. Information Collected

Table 18.1 shows the information collected so far for our running example. The first column shows the *process*. For our running example, we have focused on the *content payment* process. The second and third column indicate the type of requirement and the requirement itself collected for the process. The *functional* requirements were derived in Chapter 12[4] from the content payment process. The according *security* requirements were added in Chapter 13[5], while the *compliance* requirements were obtained in Chapter 16[6]. Note that for the compliance requirements we only discussed `R15` and the compliance requirement regarding *abroad transfer* (R15A) in detail. But there are some more compliance requirements. First of all, the consent for allowing an abroad transfer can also be requested by the media market directly when the content request is sent by the customer (*R3*). Hence, here we have an alternative how to implement the compliance requirement. Additionally, whenever the media market receives or sends private data, it has to be made sure that only private data necessary for fulfilling the purpose of the request is collected or used (*data minimization*). Additionally, no unauthorized *3rd party* is allowed to get *access* to the private data. We have also added some *cost* requirements, which express the cost targets for running and maintaining the different functions. All requirements can be found in Appendix F.1.4[7]. For each requirement we also collected the information if this requirement is a must have requirement or not (column 4). The following columns show the goals of the different stakeholders. We have collected the stakeholders in Chapter 10[8], and the goals in Section 11.1[9]. As already discussed in Section 11.1[10], the leaves of the goal trees are of

[1] Page 327
[2] The method for interaction detection is a contribution of Azadeh Alebrahim. The general idea of relaxation templates for alternative generation, the method for using them, and the templates for security and costs are a contribution of the author.
[3] Page 331
[4] Page 213
[5] Page 221
[6] Page 281
[7] Page 528
[8] Page 181
[9] Page 203
[10] Page 203

Process	Type	Requirement	Must Have	Content Aggregator: Comply to Law	Avoid Denial of Service	Avoid Data Loss	Avoid Running Costs	Avoid Infringement Costs	Avoid Maintenance Costs	Avoid Management Costs	Get Money	Sell Media	Process Information	Collect Provider Information	Collect Maximum of Information	Collect Customer Information	Customer: Pay Correct Bill	Minimize Bill	Find Content	Get Content	View Content	Media Data Not Manipulated	Bill not Manipulated	Minimize Private Data	Protect Stored Private Data	Protect Communicated Private Data	Payment Gateway Provider: Comply to Laws	Avoid Infringement Costs	Avoid Maintenance Costs	Avoid Management Costs	Avoid Data Loss	Avoid Denial Of Service	Sell Service	Get Money
Content Payment	Functional	R3 Request Content	x									x	x	x	x	x				x													
		R15 Check Payment Data	x								x																						x	x
		R16 Reject Payment Data	x								x				x	x											x	x					x	x
		R17 Accept Payment Data	x								x	x			x	x				x							x	x					x	x
		R18 Check Payment Information						x					x	x	x	x																		
	Security	SR3A Confidentiality				x	x	x	x																	x								
		SR3B Integrity			x		x	x	x			x					x					x	x			x								
		SR3C Availability	x		x		x	x	x		x	x	x	x	x	x			x	x													x	x
	Compliance	CR3D Abroad Transfer	x	x			x	x																	x		x	x						
		CR3E Data Minimization		x				x																x										
		CR3F No 3rd Party Access	x	x			x	x																	x									
		CR15A Abroad Transfer	x	x			x	x																	x		x	x						
		CR15B Data Minimization		x				x																x										
		CR15C No 3rd Party Access	x	x			x	x																	x									
	Costs	COR 3G Running Costs					x																											
		COR 3H Maintenance Costs							x																									
		COR 15D Running Costs					x																											
		COR 15E Maintenance Costs							x																									
		COR 16A Running Costs					x																											
		COR 16B Maintenance Costs							x																									
		COR 17A Running Costs					x																											
		COR 17B Maintenance Costs							x																									
		COR 18A Running Costs					x																											
		COR 18B Maintenance Costs							x																									

Table 18.1.: *Information Collected*

importance, therefore they are shown in the table. The relations between the requirements and the goals are described in the following.

R3 is a *must have* requirement because without content requests there is no business case for the content aggregator. Hence, it is related to the goal *sell media*. Of course, a content request contains information which can be *processed*, and the information contains valuable *provider information* as well as *customer information* which can be *collected*. For the customer R3 is required to *get content*.

R15 is also a *must have* requirement because it is related to the *get money* goal of the content aggregator. It is also related to the *get money* and *sell service* goals of the payment gateway provider as each request of the content aggregator is a transaction sold for the payment gateway provider.

The same applies for *R16* and *R17*. Additionally, both requirements reveal some additional information about the customer, because they give some indication about the financial situation of the customer. The feedback to a payment data check is also regulated and protected by different laws. Hence, the goals *comply to law* and *avoid infringement costs* of the payment gateway provider are touched. R17 is also related to *sell media* of the content aggregator and *get media* of the customer, because in case of a successful payment the media is sold to the customer.

R18 is *not a must have* requirement, because media could be sold without checking each content request for validity. But this would cause *infringement costs* because an invalid content request might be fulfilled which can cause trouble with customers as well as content providers. Additionally, checking the information allows to *collect* further information about *providers* and *customers*.

In the first place, *SR3A* fulfills the goal of *avoid data loss*. But it also helps to *avoid infringement* as well as *maintenance costs*, as a protected communication also lowers the odds of a manipulated request leading to trouble with the customer, or interference with the machine itself. But it causes extra *running costs*.

SR3B helps to fulfill the goal of *avoid denial of service*, because an attacker who manipulates requests makes a correctly working media market unavailable to the customer. In consequence, it also helps to *avoid infringement* as well as *maintenance costs*, as it lowers the odds of a manipulated request leading to trouble with the customer, or interference with the machine itself. But it causes extra *running costs*.

SR3C is the main requirement to fulfill *avoid denial of service*. A denial of service attack is a serious threat to the business case of the content aggregator, because in case the media market is not accessible no *selling of media* and *getting money* is possible, it is a *must have* requirement. Additionally, any *information* can only be *collected* if the service is available. Moreover, an unavailable media market might lead to sues of content providers as well as customers causing *infringement costs*, and also the recovery from an attack causes *maintenance costs*. But it causes extra *running costs*.

As already discussed in Section 16.5[11], the consequences of transferring personal data abroad without a consent are severe. Hence, *CR3D* is a must have requirement like its alternative *CR15A*. Of course, these requirements help to fulfill the goal of being *compliant with law*, and to avoid the related *infringement costs*. *CR3A* or *CR15A* also help to fulfill the goal of the customer to protect *stored private data*, because for each action which might reveal private data of the customer to others the customer has to be asked. Additionally, also the *compliance* goal and *infringement costs* goal of the payment gateway provider are related, because the payment gateway provider has to assure that data sent to and processed by it is allowed to be sent and processed. Both causes extra *running costs*. The same reasoning applies for *CR3F* and *CR15C* concerned with the access of third parties to private data.

The requirements *CR3E* and *CR15B* concerned with data minimization stress the importance and validity of the customers' goal of *minimizing private data*. Of course, the content provider introduces these requirements to be *compliant* and avoid *infringement costs*. But collecting more data than really needed does not have that sever consequences in a law case as, for example, the abroad transfer. Hence, they are not "must have" requirements.

All the cost requirements are related to the according cost goals.

Note that we do not show how to relate goals and requirements in this work. We refer to some recent papers on the matter: There are papers which reported a successful integration of goal-oriented and problem-oriented methods (Liu and Jin, 2006 [243]; Beckers, Faßbender, Heisel, and Paci, 2013 [48]; Mohammadi, Alebrahim, Weyer, Heisel, and Pohl, 2013 [270]).

18.2. Detection of Interactions

Nowadays, for almost every software system various stakeholders with diverse interests exist. These interests give rise to different sets of requirements. The combination of these sets may lead to unwanted *interactions* among the requirements. Such interactions among requirements cannot be detected easily. Unwanted interactions among different artifacts are typically referred to as feature interaction, originally identified as a problem in the domain of telecom-

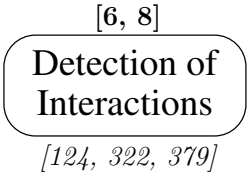


Figure 18.1.: *Detection of Interactions*

[11] Page 302

munications (Calder, Kolberg, Magill, and Reiff-Marganiec, 2003 [93]). Interactions can occur in all stages of the software development process. The analysis of interactions and dependencies among requirements is called *requirements interaction management* (Robinson et al., 2003 [322]) and is regarded as one area of requirements engineering. Robinson et al. (2003 [322]) define it as the "set of activities directed towards the discovery, management, and disposition of critical relationships among sets of requirements". The area of requirements interaction management is quite similar to the area of feature interaction. Both areas deal with discovering and resolving negative interactions. However, the area of feature interaction has a narrower view of requirements (Robinson et al., 2003 [322]).

In general, the deviation between the intended behavior and structure as formulated by single requirements of a stakeholder and the overall behavior and structure of the resulting system- or software-to-be is called requirement *inconsistency* (France and Rumpe, 2007 [149]; van Lamsweerde et al., 1998 [379]). Such inconsistencies can stem from different sources. The *first source* is the different understanding of terms and different views on the system-to-be of different stakeholders. Missing or misleading information also adds to this class of inconsistencies [149, 358]. A *second source* are inconsistencies which stem from the transformation between different kinds of representations and models (France and Rumpe, 2007 [149]). *Another important source* are interactions between requirements which lead to an unexpected behavior. For functional requirements this source is already known as feature interaction for a long time, e.g. in the domain of telecommunication (Calder et al., 2003 [93]; Cameron and Velthuijsen, 1993 [94]; van Lamsweerde et al., 1998 [379]). For interactions, one can distinguish between unwanted and desirable interactions. The strongest type of interactions are conflicts in which requirements deny each other, and dependencies where one requirement can be only fulfilled when another requirement is also fulfilled. Between these extrema, there are different shades of negative or positive influences (Robinson et al., 2003 [322]; van Lamsweerde et al., 1998 [379]). For this chapter, we assume that inconsistencies in terms of the first and second source are solved and we will focus on conflicts.

In Alebrahim, Faßbender, Heisel, and Meis (2014 [8]), we presented a method which allows one to find those functional requirements, which are modeled as problem diagrams, which might be in conflict with each other. The purpose of this method is not to prove the actual conflicts, but to remove all the combinations of functional requirements which cannot conflict. The remaining requirements have to be analyzed in detail using other methods. We have shown that our method is able to reduce the set of candidates by more than 90%, which significantly lowers the effort of applying the detailed interaction analysis. In this thesis we are focusing more on quality requirements, and the running example does not contain any functional requirements interaction[12]. We refer the interested reader to Alebrahim, Faßbender, Heisel, and Meis (2014 [8]) for more details.

In the following, we describe briefly a method to detect candidates for negative interactions based on pairwise comparisons between quality requirements. The full details can be found in Alebrahim, Choppy, Faßbender, and Heisel (2014 [6]). The reason for only giving an overview is that this particular method is the main contribution of Azadeh Alebrahim. The process is described in the following.

To restrict the number of comparisons, we perform a *preparation phase*, in which we investigate which two types of quality requirements may be in conflict in general. In doing so, we consider different types and according sub-types of quality requirements. For example, we consider the sub-types of a security requirement, namely integrity, confidentiality, and authenticity as types. The preparation phase results in a table containing all types of quality requirements to be considered. We compare each two types of quality requirements regarding potential conflicts.

[12]We applied the method as described in Alebrahim, Faßbender, Heisel, and Meis (2014 [8]), but the example was crafted that carefully that the set of requirement conflict candidates was empty in the end.

	Confidentiality	Integrity	Authenticity	Performance	Privacy	Costs
Confidentiality	-	-	x	x	-	x
Integrity	-	-	-	x	-	x
Authenticity	x	-	-	x	-	x
Performance	x	x	x	x	x	x
Privacy	-	-	-	x	-	x
Costs	x	x	x	x	x	x

Table 18.2.: *Possible Interactions among Types of Quality Requirements in General*

If conflicts are possible, we enter a cross in the cell, where the two quality requirements cross, otherwise a minus. For example, no interactions between a confidentiality requirement and a privacy requirement are expected. Therefore, the cell crossing these two requirement types in the table contains a minus. In contrast, a confidentiality requirement might be in conflict with a performance requirement. Hence, the corresponding cell contains a cross. Table 18.2 shows possible interactions among security (confidentiality, integrity, authenticity), performance, privacy, and cost requirements in general.

Interactions among quality requirements of different types can occur either between quality requirements related to the same functional requirement or among those related to different functional requirements. We classify quality requirements and their relations to the functional requirements into four cases, see Table 18.3. Case one arises when we consider two quality requirements of the same type related to the same functional requirement. The second case is concerned with considering two quality requirements of different types that are related to the same functional requirement. Case three occurs when two quality requirements of the same type are related to different functional requirements which must be fulfilled in parallel. In the fourth case, two quality requirements of different types are related to different functional requirements which must be fulfilled in parallel. We treat each case in a separate phase in our method. The result of this classification is represented in Table 18.3. The abbreviations FR and QR stand for "Functional Requirement" and "Quality Requirement", respectively.

The result of applying the method is shown in Table 18.4. It shows the requirements as rows as well as columns. In consequence, a cell given by a particular column and row shows the relation between the requirements at hand. A ***D*** indicates that the requirement given in the row is dependent on the requirement given in the column. A ***C*** indicates that the requirement given in the row complements the requirement given in the column. An ***A*** indicates that the requirements are alternatives to each other. A *red flash* indicates that these requirements are in conflict and deny each other. A *red downwards arrow* indicates that the requirement given in the row complicates the fulfillment of the requirement given in the column. In turn, a *green upwards arrow* indicates that the requirement given in the row eases the fulfillment of the requirement given in the column.

In the following, we will only discuss the conflicts, negative influence, positive influence relations, because the dependent on and complements relations are quite trivial. *SR3A* of course

	FR, type of QR	Condition	Row in QR table	Method's phase
Case 1	same FR, same type of QR	-	rows related to same FR in same QR table	phase 1
Case 2	same FR, different types of QR	-	rows related to same FR in different QR tables	phase 2
Case 3	different FR, same type of QR	in parallel	rows related to different FR in same QR table	phase 3
Case 4	different FR, different types of QR	in parallel	rows related to different FR in different QR tables	phase 4

Table 18.3.: *Classification Table*

Type		Functional					Security				Compliance							Costs									
	Requirement	R3 Request Content	R15 Check Payment Data	R16 Reject Payment Data	R17 Accept Payment Data	R18 Check Payment Information	SR3A Confidentiality	SR3B Integrity	SR3C Availability	CR3D Abroad Transfer	CR3E Data Minimization	CR3F No 3rd Party Access	CR15A Abroad Transfer	CR15B Data Minimization	CR15C No 3rd Party Access	COR 3G Running Costs	COR 3H Maintenance Costs	COR 15D Running Costs	COR 15E Maintenance Costs	COR 16A Running Costs	COR 16B Maintenance Costs	COR 17A Running Costs	COR 17B Maintenance Costs	COR 18A Running Costs	COR 18B Maintenance Costs
Functional	R3 Request Content																								
	R15 Check Payment Data	D																							
	R16 Reject Payment Data		D																						
	R17 Accept Payment Data		D																						
	R18 Check Payment Information	D																							
Security	SR3A Confidentiality	C										▲				↯ ▼									
	SR3B Integrity	C														▼	▲								
	SR3C Availability	C														↯	▲								
Compliance	CR3D Abroad Transfer	C											A			▼									
	CR3E Data Minimization	C										▲													
	CR3F No 3rd Party Access	C																							
	CR15A Abroad Transfer		C							A								▼							
	CR15B Data Minimization		C												▲										
	CR15C No 3rd Party Access		C																						
Costs	COR 3G Running Costs	C					↯		↯																
	COR 3H Maintenance Costs	C																							
	COR 15D Running Costs		C																						
	COR 15E Maintenance Costs		C																						
	COR 16A Running Costs			C																					
	COR 16B Maintenance Costs			C																					
	COR 17A Running Costs				C																				
	COR 17B Maintenance Costs				C																				
	COR 18A Running Costs					C																			
	COR 18B Maintenance Costs					C																			

↯ Conflict ▼ Negative Influence ▲ Positive Influence
D Depends C Complements A Alternative

Table 18.4.: *Relations between Requirements*

eases the protection against third parties, but it is only aimed at attackers. But in the sense of the privacy law, protecting against third parties is not only a protection against attackers, but any counter party which is not explicitly allowed to get the data. Hence, *SR3A* does not completely cover *CR3F*, because it, for example, does not protect the data against selling them by the content aggregator. *SR3A* is in conflict with the running costs allowed for R3 (*COR3G*), because the media market is supposed to run as service on a cloud platform, and each CPU cycle caused by a service leads to additional costs. An encryption solution, for example, would therefore generate to much additional cycles to meet the costs requirement. Note that we assume that the running costs requirement is formulated that way that only a small number of CPU cycles are left for other tasks than executing the pure functionality. *SR3A* also influences the maintenance costs, because, for example, network traffic cannot be analyzed that easily anymore.

In turn, *SR3B* can be fulfilled along with *SR3G*, but complicates the fulfillment of the cost target, but it eases to fulfill the maintenance costs requirement *SR3H*, because the integrity checks allow to trace problems to their origin more easily.

The availability target of 99.99% as given by *SR3C* cannot be fulfilled along with the running costs requirement (*COR3G*), because such an availability level is only guaranteed by the cloud provider for a greater price. But as each case of unavailability causes maintenance costs, *SR3C*

eases the fulfillment of *COR3H*.

The fulfillment of the requirements related to the abroad transfer (*CR3D* and *CR15A*), which requires additional functionality for getting an explicit consent by the customer, of course generated extra running costs which complicates the fulfillment of the according running cost requirements (*COR3G* and *COR15D*). Remember that *CR3D* and *CR15A* are alternative requirements which is indicated be the **A**.

The data minimization requirements (*CR3E* and *CR15B*) of course help to avoid that private data is accessed by third parties, because the less private data is used, the less is the probability that someone wants to possess them.

After this step, the relations between the requirements as originally required to solve the full problem at hand are known.

18.3. Generation of Alternatives

To enable the selection of a final set of requirements, which is as near to the optimal solution for every stakeholder as possible, we need to generate alternatives for the problematic requirements. An original requirement might be excluded from the final set, but a weaker variant of this requirement might be included in the optimal set of requirements. For example, for security requirements there can be certain kinds of attackers we want to be secured against. However, we may not be able to address a strong attacker with certain properties such as the given time and resource limits. Hence, we propose a method for relaxing such properties in order to generate alternatives for problematic requirements. Generated alternatives are used as recommendations for stakeholders. Those alternatives that are not acceptable for stakeholders are excluded before they are used as input for the next step of the method. Figure 18.3 illustrates our method.

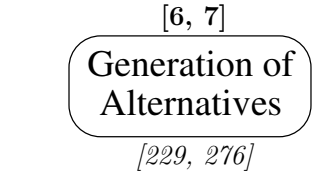


Figure 18.2.: *Generation of Alternatives*

Based on the type of requirement we want to generate alternatives for, there are different properties which might be relaxable. As the qualities addressed by different requirements are

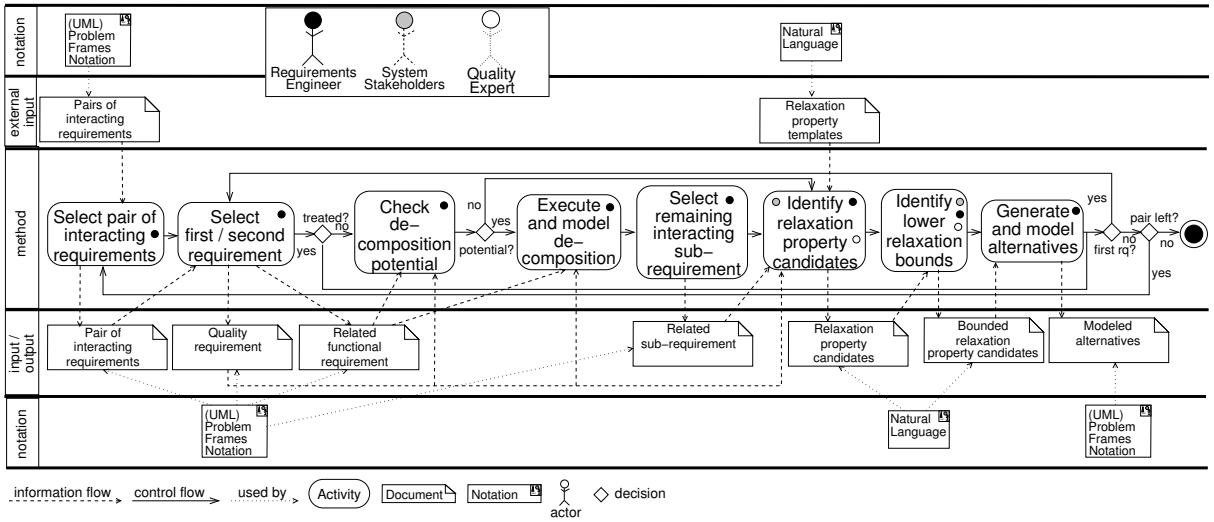


Figure 18.3.: *Method for Alternative Generation*

very different, so are the properties which can be used to relax a requirement. But for a particular kind of quality those properties are the same. Hence, it is possible to define a property template for each quality, which can be instantiated for a requirement belonging to this quality. For each quality, we capture the following information in the template (see Tables 18.5 and 18.6): *Property* describing the quality-relevant properties, *Possible Values* describing the range of values the property can take, *Value Original Requirement* representing the value of property for the original requirement before relaxing, *Upper/Lower Bound*, describing the lower or upper bound (depending on the property) each property can take when relaxing, *Value R* representing the values of the relaxed properties for requirements alternatives. In the following, we present the templates for the qualities security and cost, before we introduce our method for generating alternatives in detail.

18.3.1. Relaxation Template for Security

For security it is the type of attacker that influences the restrictiveness of a security requirement. How much resources and effort have to be spent on a security requirement, how much influence the requirement has on the behavior of the overall system-to-be, and which solution has to be chosen later on for fulfilling the requirement, depends on the abilities of the attacker. While it is almost impossible to secure a system against an almighty attacker, defending against a layman (see (International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), 2009 [195])) can be easily achieved without a big impact on the rest of the system.

To describe the attacker we use the properties as described by the Common Methodology for Information Technology Security Evaluation (CEM) (International Organization for Standard-

Quality: Security, Requirement SR3A, Alternatives SR3I, SR3J, SR3K	Property (CEM)	Possible Values	Value Original Requirement	Upper/Lower Bound	Value SR3I	Value SR3J	Value SR3K
	Preparation time	one day, one week, two weeks, one month, two months, three months, four months, five months, six months, more than six months	more than six months	one month	four months	two months	one month
	Attack time	one day, one week, two weeks, one month, two months, three months, four months, five months, six months, more than six months	more than six months	one month	more than six months	three months	one month
	Specialist expertise	laymen, proficient, expert, multiple experts	multiple experts	proficient	multiple experts	expert	proficient
	Knowledge of the TOE	public, restricted, sensitive, critical	public	public	public	public	public
	Window of opportunity	unnecessary / unlimited, easy, moderate, difficult	easy	easy	easy	easy	easy
	IT hardware/software or other equipment	standard, specialized, bespoke, multiple bespoke	multiple bespoke	bespoke	multiple bespoke	multiple bespoke	bespoke
	Availability Specific						
	Degree of Availability	0% - 100%	-	-	-	-	-

Table 18.5.: *Security Relaxation Template and its Instantiation for SR3A*

ization (ISO) and International Electrotechnical Commission (IEC), 2009 [195]) for vulnerability assessment of the TOE (target of evaluation, for example, our system-to-be). How to integrate this attacker description into problem frames is described in Hatebur and Heisel (2009 [173]), and Hatebur and Heisel (2010 [174]). The properties to be considered (according to CEM) are (International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), 2009 [195]):

Elapsed time "Elapsed time is the total amount of time taken by an attacker to identify a particular potential vulnerability ..., to develop an attack method and ... to mount the attack ...." We distinguish between the *preparation time* and the *attack time*.

Specialist expertise "Specialist expertise refers to the level of generic knowledge of the underlying principles, product type or attack methods ...."

Knowledge of the TOE "Knowledge of the TOE refers to specific expertise in relation to the TOE."

Window of opportunity "Identification or exploitation of a vulnerability may require considerable amounts of access to a TOE that may increase the likelihood of detection. ... Access may also need to be continuous, or over a number of sessions."

IT hardware/software or other equipment "... the equipment required to identify or exploit a vulnerability."

An additional property not mentioned in (International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), 2009 [195]) is specific to availability requirements. Here, we can also relax the degree to which the system shall be available.

The resulting relaxation template is shown in Table 18.5.

18.3.2. Relaxation Template for Costs

For costs, the relaxation table is quite simple (see Table 18.6). There are four attributes which can be relaxed. First, the amount of money which is paid for a functionality within a defined time span irrespective its usage, so called *fix costs*. Second, the money to be spent for each execution of a functionality, so called *variable costs*. Third, the amount of working hours spent on a functionality within a time span regardless the number of executions (*Fixed effort*). Fourth, the amount of working hours spent on every execution of a functionality (*Variable effort*).

Quality: Costs, Requirement COR3G, Alternatives COR3L, COR3M, COR3N	Property	Possible Values	Value Original Requirement	Upper/Lower Bound	Value COR3L	Value COR3M	Value COR3N
	Fix costs	amount of money per time period	0 €/ month	0 €/ month	0 €/ month	0 €/ month	0 €/ month
	Variable costs	amount of money per execution	0.01 €	0.02 €	0.014 €	0.018 €	0.02 €
	Fixed effort	working hours per time period	0 hours / month	0 hours / month	0 hours / month	0 hours / month	0 hours / month
	Variable effort	working hours per execution	0 hours	0 hours	0 hours	0 hours	0 hours

Table 18.6.: *Cost Relaxation Template and its Instantiation for COR3G*

18.3.3. Application of the Method for Generating Alternatives

In the following, we describe our method to generate alternatives for interacting requirements to be used in further steps(see Fig. 18.3).

1. **Select pair of interacting requirements** Table 18.4 is the input for the generation of alternatives. We have to analyze each pair for possible alternative requirements, which resolve or relax the interaction.
   For our example, we select the requirements pair SR3A and COR3G as an example.
2. **Select first / second requirement** For the selected pair, we have to check each of the two requirements for possibilities to resolve the interaction. Hence, we have to execute the next steps for both requirements.
   Both requirements provide the possibility to be relaxed in order to resolve the interaction. Hence, we perform the next steps for both requirements SR3A and COR3G. In Table 18.5 and Table 18.6, we fill the column "value original requirement" for these two requirements. Regarding the confidentiality, we state that the system must be protected against the strongest attacker. Hence, we select for each property the strongest one to obtain values for original requirement SR3A. Regarding the costs, the content aggregator wants to use a fully flexible cloud payment model, which means that the content aggregator pays per CPU cycle / RAM usage / storage usage, and no basic fee. The functionality of R3 does not involve any activities of the staff of the content aggregator. Hence, the only property which has a value is "variable costs".
3. **Check decomposition potential of requirements** In the case of a complex requirement, it might help to separate the source of interaction from the rest of the requirement. The separated requirements can be treated differently. It might happen that an interaction would lead to a rejection of the whole, complex requirement. In contrast, for the decomposed set of requirements, some parts of the original requirement might remain in the solution.
   The quality requirements SR3A and COR3G complement the functional requirement R3, which is concerned with accepting the content request of a customer. This is not a complex problem (see Figure 13.2[13]) and cannot be decomposed further.
4. **Execute and model decomposition** If a decomposition is possible, it has to be executed, and the result has to be modeled.
   R3 cannot be decomposed.
5. **Select remaining interacting sub-requirements** In case of a decomposition, only the sub-requirement, which is the source of the interaction, has to be analyzed further.
   We did not decompose anything.
6. **Identify relaxation property candidates** Based on the type of requirement, there are different properties, which are candidates to be relaxed. These candidates are fixed for each kind of requirement. Hence, we can use predefined templates to identify these properties. For each property the actual value regarding the interacting requirement has to be stated. Next, it has to be decided if this value for the property is a hard constraint, which cannot be changed, or a soft constraint, which might be relaxed. In the second case, we identified a relaxation candidate. Up to this point the requirements engineer could work alone, but for identifying properties which might be relaxed, he / she needs the decisions of the system stakeholders. Hence, they have to be involved in this step. Additionally, having an expert for the quality at hand might be helpful, as for some qualities, such as security, it needs some expertise to decide what can be relaxed, and which properties have to be fixed. In consequence, this step and the following one involve the requirements engineer, system stakeholders, and quality experts.

[13] Page 222

For the security requirement SR3A (Table 18.5), we figure out that the properties "knowledge of the TOE" and "window of opportunity" are fixed and cannot be relaxed, because the specification of the external interface of the media market has to be known to allow others to connect to it. This also implies that the access to the media market is accessible for everyone. The rest of properties can be relaxed to generate alternatives for the original requirement SR3A. For COR3G (Table 18.6) the only property which can be changed are the variable costs.

7. **Identify upper/lower relaxation bounds** For each property the upper/lower bound which is still acceptable has to be identified. The upper/lower bounds of all properties form the worst case scenario, which is still acceptable for a requirement.
   To identify "upper/lower bounds" for the requirement SR3A, we assume that the system has to be protected at least against an attacker, who is "proficient", and has "one month" for preparing the attack, and has "one month" for the attack itself, and has a "bespoke" equipment for performing the attack. Hence, we are able generate different alternatives reaching from very mighty attackers to rather poor attackers. For COR3G (Table 18.6), we decide that up to the double amount of variable costs will not break the business model of the content aggregator.
8. **Generate and model alternatives** The first alternative is the requirement realizing the worst case scenario. Between the original requirement and this lower bound requirement, several other requirements can be generated by varying the relaxation candidates. For each generated requirement it has to be checked whether it eliminates the interaction or not. If it does not, further relaxation is needed. The generated alternatives have to be modeled. For modeling alternatives, there is also a profile available which allows one to treat variability in problem diagrams (Alebrahim, Faßbender, Filipczyk, Goedicke, Heisel, and Konersmann, 2014 [7]).
   To relax the properties and thus generate alternatives for the requirements SR3A and COR3G, we choose values between the "value original requirement" and "upper/lower bound" for properties that can be relaxed. Relaxing possible properties results in requirements alternatives SR3I, SR3J, and SR3K for the original requirement SR3A and in requirements alternatives COR3L, COR3M, and COR3N for the original requirement COR3G. In this way, we cannot say that we assuredly resolve interactions between quality requirements, but we can weaken them for sure or even resolve them ideally.

In the same manner we tread SR3C (see Table F.44[14]). Afterward, we have all our alternatives in place.

18.4. Conclusion

In this chapter, we have

- given a consolidated overview of information collected so far,
- discussed the matter of interactions between requirements,
- outlined methods to detect interactions,
- discussed the importance of alternatives for resolving conflicts,
- and presented a guided method for generating alternatives for quality requirements using so called requirements relaxation templates.

As we applied all those methods to our running example, we now have a detailed view on the interactions between particular requirements, and which alternatives exist for the conflicting requirements.

[14] Page 541

Chapter 19

Valuation of Requirements

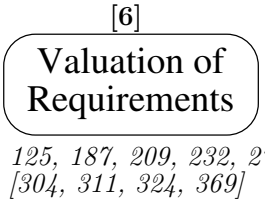


Figure 19.1.: *Valuation of Requirements*

For an informed requirements selection, the requirements need (a) value(s) on which a requirements engineer can base his / her decision. This can be achieved by prioritization, or, for example, cost estimations, or empirical data such as, for example, customer surveys. The valuating measure or method has to be selected in this step and the values for the requirements have to be elicited. Furthermore, the valuation has to be documented for later use. For this chapter, we have chosen the ANP as valuation method. We discuss this decision in Section 19.1. Then, we show how to set up an ANP network for the information collected so far (Section 19.2). Afterward, we show the results of applying our ANP solution to our running example in Section 19.3[1]. With Section 19.5[2] we conclude this chapter. This chapter is based on Alebrahim, Choppy, Faßbender, and Heisel (2014 [6])[3].

19.1. Selecting a Valuation Method

In Section 1.4.2[4], we already discussed aspects which are of importance for requirements selection in general. Some of them are of particular importance for selecting a valuation method, such as inconsistent judgments, absence of quantitative data, multidimensional values, decision making as a collaborative process, ease of use, and tool support. Based on these aspects, we took a look at several valuation methods. For valuating and comparing alternatives among each other, there are several methods known, such as direct scoring (Pomerol and Barba-Romero, 2000 [315]), Even-Swaps (Mustajoki and Hämäläinen, 2007 [275]), win-win negotiating (Ruhe et al., 2002 [324]), the analytic hierarchy process (Saaty, 2005 [327]; Saaty and Ozdemir, 2003 [330]), the analytical network process (Saaty, 2008 [328]; Saaty, 2005 [327]), and so forth. They all support the valuation or ranking of alternatives by either eliminating alternatives successively, or comparing and ordering them. We decided to use ANP for the valuation of requirements. There are several reasons for this decision:

Capture the complexity of the decision problem As we see from the goal model (Figure 11.2[5]) and the relations between goals, requirements, and requirement alternatives (Table 18.1[6] and Table 18.4), we have a very complex decision problem. Even the decision about the

[1] Page 337
[2] Page 339
[3] The valuation method is a contribution of the author.
[4] Page 15
[5] Page 204
[6] Page 322

value or rank of a requirement is not straightforward, because there are so many goals a particular requirement might contribute to. ANP allows one to model the decision problem coherently to the real problem as described and modeled in the preceding steps. For AHP, Even Swaps, or simple ranking, simplifications would be needed, making the outcome unreliable (Saaty, 2008 [329]).

Reduce the complexity of decisions In ANP decisions are reduced to pairwise comparison. This has been proven to be the most natural decision a person can make (Saaty, 2005 [327]; Karlsson et al., 1998 [210]).

Coping with fuzzy values ANP does not require to give concrete numbers for the value of a requirement or goal, but relies on relative comparisons (Saaty, 2005 [327]). This is an important property as giving fixed numbers can be hardly achieved in the early phases of software engineering. Furthermore, ANP has proven to be one of the most reliable decision techniques for fuzzy environments (Saaty, 2008 [329]).

Detecting and handling inconsistencies ANP allows one to compute and check the consistency of the different comparisons. Thus, inconsistencies can be avoided. However, ANP even works for comparisons with small-scale inconsistencies (Saaty, 2005 [327]).

Merging of different views and dimensions for a decision ANP allows one to merge results for different dimensions, like benefits and costs, of a decision. Furthermore, it is easy to integrate different views of different stakeholders (Saaty and Ozdemir, 2003 [330]; Saaty, 2005 [327]).

Tool support For ANP, there are different support tools.[7]

19.2. A Method to Use ANP

Up to this point, we covered only step 1 of ANP as described by Saaty (2008 [329]) (Section 2.2.5[8]) which is about collecting all the necessary information. The steps shown in Figure 19.2 for setting up the ANP-Network (see sketch in Figure 19.3), covering steps 2 to 4 of ANP, are as follows:

1. **Set up top-level network**. For the top-level we set up the goal cluster containing the goal “Overall best system’. The goal cluster is influenced by the control criteria cluster.

[7] e.g. Superdecisions (http://www.superdecisions.com/) or ANPSolver (http://kkir.simor.ntua.gr/anpsolver.html)
[8] Page 37

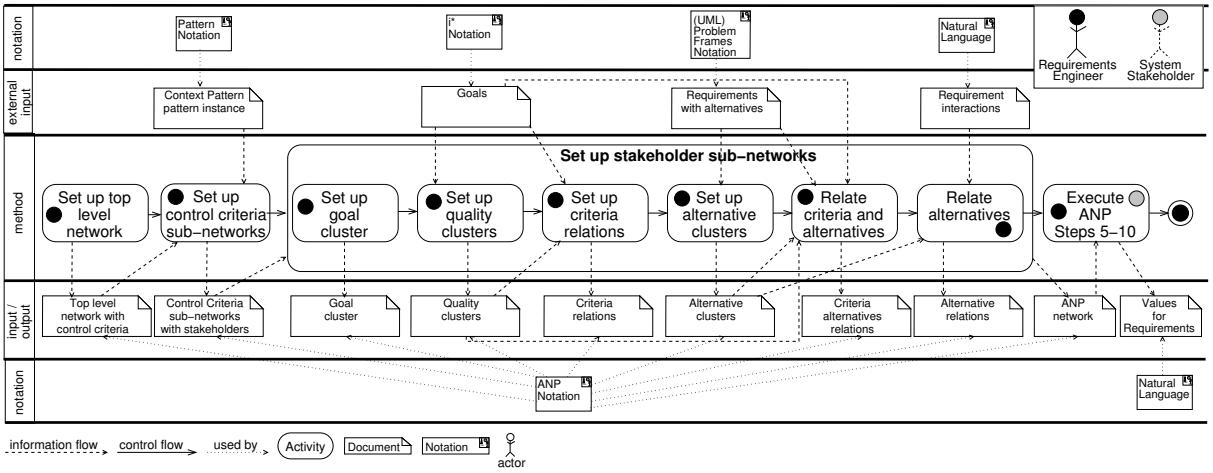


Figure 19.2.: *Method for valuating requirements*

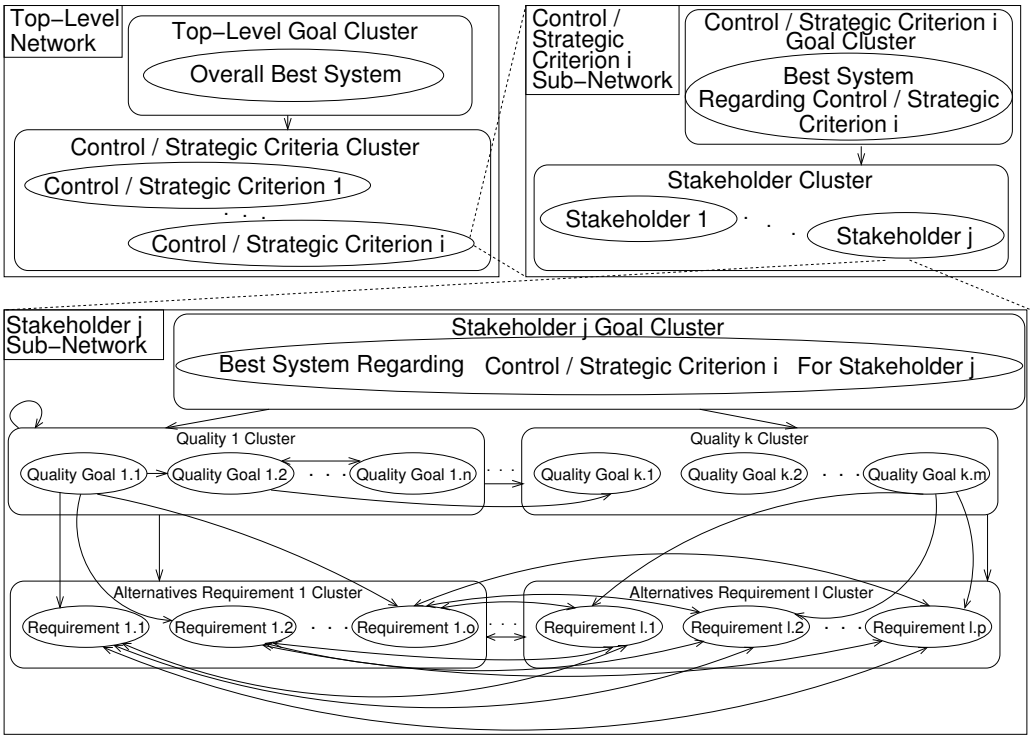


Figure 19.3.: *The ANP-Network*

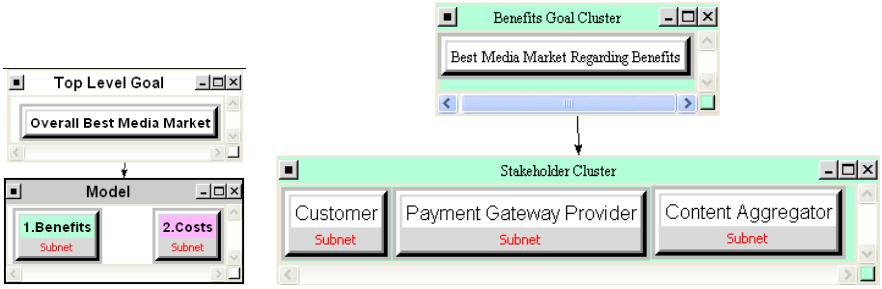


Figure 19.4.: *Top Level Network (left) And Control Criterion Sub-Network for Benefits (right) modeled in Superdecisions*

For the control criteria, we stick to the Benefits, Opportunities, Costs, and Risks (BOCR) criteria as suggested by Saaty (Saaty and Ozdemir, 2003 [330]). But nevertheless, it is possible to choose other strategic criteria here. Strategic criteria do not influence each other. Hence, for this network we have a hierarchy from the goal to the criteria (see Figure 19.3 on upper left-hand side). *For our example, we decide to use only benefits and costs (see Figure 19.4 left-hand side).*

2. **Set up control criteria sub-networks** For each control criterion we add a sub-network. A sub-network consists of a goal cluster with a goal like "Best system regarding benefits". The goal cluster is influenced by a stakeholder cluster, which contains a node for each stakeholder of the system-to-be. We assume the stakeholders to be independent, because the goals a stakeholder wants to achieve are not based on the perception of these goals by other stakeholders. Thus, we do not have any inner dependence influence. *For our example, we have the three stakeholders customer, content aggregator, and payment gateway provider*

(see Figure 19.4 right-hand side).

3. **Set up stakeholder sub-networks** The stakeholder sub-networks are the real ANP-networks, while the top-level and control-criteria level just serve for the integration of dimensions and views on the system-to-be. Hence, we split up the setup of stakeholder sub-networks into some sub-steps. Note that these steps directly apply for the benefits criterion. For other criteria they might have to be modified. For risk and opportunity the steps can be performed without modifications, but for costs we removed the quality clusters and only introduced the clusters "development costs", and "operative costs". *The resulting stakeholder sub-network for the Customer regarding benefits is shown in Figure 19.5. The sub-networks for the content provider and payment gateway provider are shown in Appendix F.1.7*[9].

   a) **Set up goal cluster** Add a goal cluster containing a goal like "Best system for stakeholder A regarding control criterion C". All top-level goals and their satisfaction level have an influence on the overall goal. Hence, the goal cluster is influenced by all other clusters except the alternative clusters. *For our example, the top-level goals economy, fun, security, and privacy have an influence on the overall goal "Best Media Market for Customer regarding Benefits" (see Figure 19.5).*

   b) **Set up quality cluster with criteria** For each top-level goal, such as security, economy, or privacy, set up a cluster. Each cluster contains the leaves (hard goals / soft goals / tasks) of the goal model for a stakeholder as nodes. A cluster only contains those leaves which are a result of the decomposition of the corresponding top-level goal using and / or / makes / decomposition relations. *For example, for the top-level goal "privacy" there is a decomposition path via "private data not disclosed", and "other parties don't get data" to "protect communicated private data" (see Figure 11.2*[10]*). Hence, we add the node "protect communicated private data" to the privacy cluster (see Figure 19.5).*

   c) **Set up node relations** For each helps / hurts relation of the goal model, add an influence relation between the corresponding nodes. Note that those goal relations are propagated down transitively from a parent goal, which is the target, to its sub-goals. *We have to relate the privacy node "protect communicated private data" with the security nodes "bill not manipulated", and "media data not manipulated", (see Figure 19.5), because the goal "protect communicated private data" helps the top-level goal "security" (see Figure 11.2*[11]*).*

   d) **Set up alternative clusters** For each requirement, add a cluster containing the alternatives for this requirement as nodes. Note that for some ANP tools it is not allowed to have several clusters for alternatives. In this case, merge them to one. This does not influence the outcome, only the readability and comprehensibility. *Figure 19.5 shows the alternative cluster (The cluster at the bottom). For superdecisions, the tool we use, it is the case that only one alternative cluster is allowed.*

   e) **Relate criteria and alternatives** Relate the criteria of the quality clusters with the requirements which influence their fulfillment. *Based on Table. 18.1*[12]*, we relate "pay correct bill" with "SR3B" and its alternatives, "Find Content" with "SR3C" and its alternatives, "Get Content" with "R3", "R17", and "SR3C", and so forth.*

   f) **Relate alternatives** Relate the alternatives with other alternatives, which have an impact on them. The alternatives for a requirement have to be related according

[9] Page 541
[10] Page 204
[11] Page 204
[12] Page 322

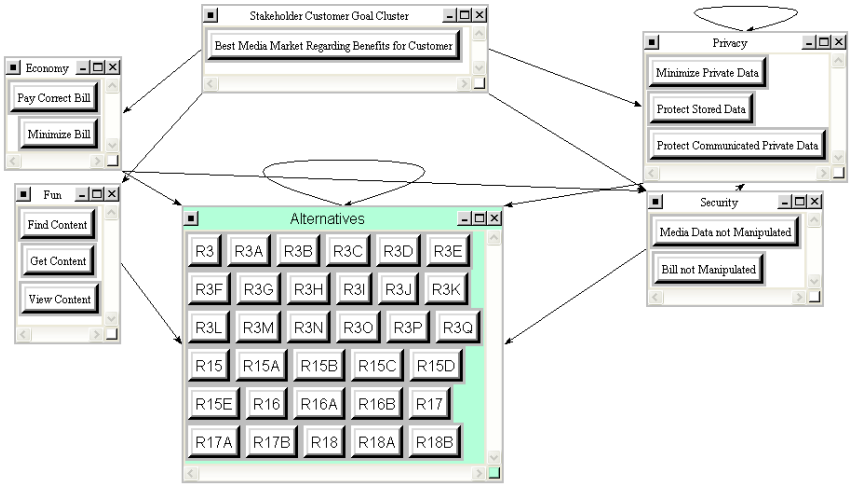


Figure 19.5.: *Stakeholder Customer Sub-Network modeled in Superdecisions*

to the relations of the original requirement. *According to Table 18.4*[13]*, we have to relate, for example, "SR3A" and "COR3H".*

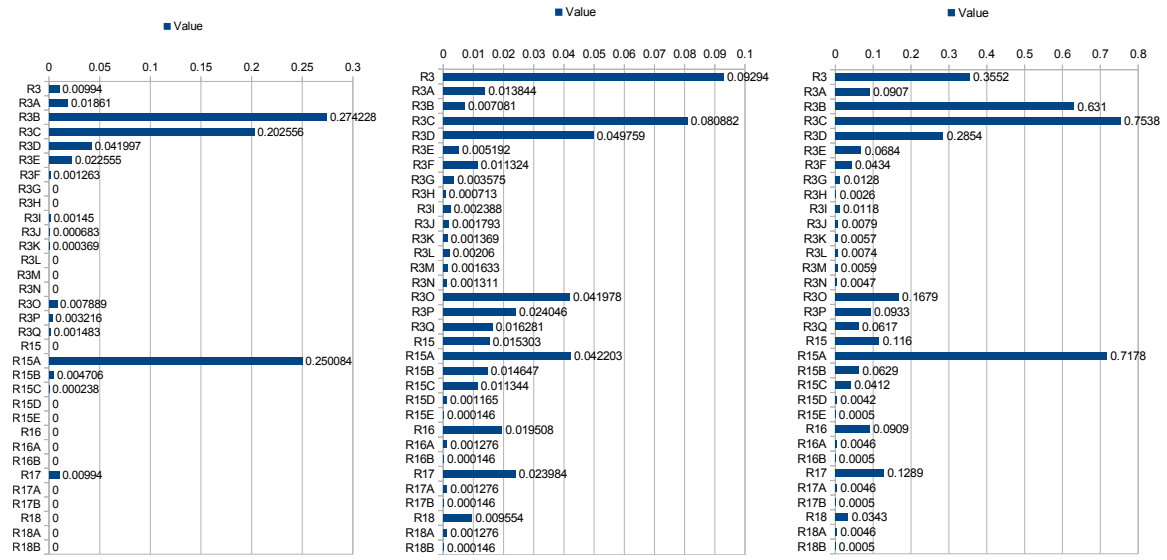
When the ANP-Network is set up, one can proceed with the regular ANP process starting with step 5 as described in Section 2.2.5[14].

19.3. Results of applying ANP

The results of our valuation for the running example are shown in Figure 19.6. From the rankings we see that the consumer prefers to have a high level of integrity, which is indicated by high value for R3B (see Fig. 19.6(a)). As a trade-off, the customer is willing to sacrifice some availability (R3C). Confidentiality as such is not rated that high by the customer (R3A) with an exception regarding his / her privacy (R15A). In contrast, the content aggregator sacrifices integrity (R3B) for availability (R3C), because the availability is of great importance for the business success of the Media Market. It might be astonishing that the content aggregator prefers to be protected against the strongest attacker, even though this causes extra costs. Here, we see the effect of the goals of being compliant and the related infringement costs goal, because the infringement costs are considered to be more severe than an increase in running / maintenance costs. But we should keep in mind that for the running costs requirement, the content aggregator still prefers R3G, which is the most strict, above the alternatives R3L, R3M, and R3N. As we already stated, R3G is in conflict with R3C. Hence, the one or the other requirement must be relaxed in the end. At this point, we clearly see that the valuation does not resolve conflicts, because the stakeholders make comparisons related to an ideal situation in which all requirements can be fulfilled at once.

These different views are then aggregated into one value for each requirement using the different levels of the ANP-Network (see Figure 19.3). First, the different views of the stakeholder regarding one control-criterion are aggregated using, e.g. the "Control Criterion Sub-Network for Benefits" (see Figure 19.4 right-hand side). Then, the different control criteria are aggregated using the top level network (see Figure.19.4 left-hand side). The aggregation of the different views of customer, content aggregator, the payment gateway, and the two strategic criteria benefits and costs, is shown in Figure 19.6(c). We see that some balancing has happened. As all

[13] Page 326
[14] Page 37



(a) Values for Rs regarding customer and benefits

(b) Values for Rs regarding content aggregator and benefits

(c) Values for Rs overall

Figure 19.6.: *ANP Result computed using superdecisions*

stakeholders see a high value in availability (R3C), it is the requirement with the top priority in the end. Also integrity (R3B) now gets a high value, because when balancing the stakeholders, we have decided to give the greatest importance to the customer, because in the end the media market is only successful when many customers are convinced to use it. We can conclude from these valuation results that it is overall still not obvious which requirements to include in the final selection, because the initial requirement always has the highest value compared to its alternatives. But the initial requirements cannot all be selected at the same time. Hence, the decision which requirement to relax is still open. The only clear cut decision is visible for getting a consent in case of an abroad transfer. Here, the preferences lean clearly towards R15A over R3D.

19.4. Related Work

Karlsson and Ryan (1997 [209]) propose a very simple AHP model for valuating requirements with regards to somehow quantified benefits and costs. Beside using the AHP and not ANP, which is a already a drawback, they do not explain how to get the actual benefit value and they require precise cost values. Hence, their solution is limited to situations in which the benefits and costs are already known, and only these particular values are of concern.

Bagheri et al. (2010 [29]) introduce a method for feature selection in software product lines. The method integrates the analytic hierarchy process (AHP) which is a simplification of the ANP. They explain how to set up an AHP model based on a feature tree, and a given set of high level goals. An interesting point in their work is that they modify the AHP in a way the number of comparisons is reduced using the different levels within the feature tree. They claim to reach a reduction of more than 60%. While the method of Bagheri et al. (2010 [29]) is limited to product lines, uses only the AHP which requires a simplification of the "real word", and does not take into account as much information as we do, the idea of introducing heuristics to lower the number of comparisons is a promising idea for future work for our method.

A method alike which aims at lowering the effort taken for using AHP is proposed by Palma

et al. (2011 [304]). It is based on the relations (conflicts, dependency, and so forth) between the requirements, and aims at finding the set of requirements which is implementable with respect to the relations. Therefore, Palma et al. (2011 [304]) use a SAT solver to find those requirements which are the reason for the infeasibility of an implementation of the initial set of requirements. Only these requirements are compared to decide which requirements to remove until an implementable set of requirements is reached. The solution of Palma et al. (2011 [304]) uses a very basic AHP in which the requirements are compared based on their relations, but all the other information we identified to be important is not taken into account. Moreover, in the end one does not get a value for each requirements, but only a set of implementable requirements, because only a small set of requirements is actually compared. Unfortunately, Palma et al. (2011 [304]) do not discuss the optimality of this set. Nevertheless, the idea of only comparing those requirements which have to be balanced might be worth a look in the future.

Elahi and Yu (2011 [125]) introduce a method for selecting alternatives given as alternative tasks within a goal model. Hence, the proposed solution is on a higher level of abstraction than our ANP-based method, which also uses goals, but also relates concrete requirements to them. The method relies on even swaps. This has several drawbacks which is also indicated by Elahi and Yu (2011 [125]). First, the result of even swap is not deterministic as the preferences of the stakeholders are never clearly stated. Hence, the order in which alternatives are compared influences the selection in the end. Second, the swaps are not easy to do because when comparing two alternatives one has to keep in mind all the goals they are related to, and as the number of swaps increase with 5^m (Elahi and Yu (2011 [125])), the effort to be taken becomes infeasible even for small systems.

Moisiadis (2002 [271]) proposes a replacement for AHP, which directly integrates stakeholders and goals and relates them to requirements. On the first view, the solution of Moisiadis (2002 [271]) looks quite similar to ours, because stakeholders and their goals, as well as relations between requirements are directly modeled. But as it is a replacement for AHP, it is not possible to relate elements on the same level. Moreover, Moisiadis (2002 [271]) provides no evidence that the calculations used to compute the final value produce reasonable values. There is no discussion on the mathematical foundations of his calculation, as well as any experiment, or alike, which might give evidence for reasonable results.

Perini et al. (2013 [311]) propose a promising method (CBRank) which uses machine learning techniques to lower the number of pairwise comparisons to be done by the user. The experiment results shown by Perini et al. (2013 [311]) indicate that CBRank produces as good rankings as the application of AHP while the number of pairwise comparisons is lowered significantly. Hence, it is part of the future work to investigate if CBRank is also able to compete with our more complex ANP while lowering the effort.

19.5. Conclusion

In this chapter, we have

- motivated our decision for ANP as valuation method,
- shown how to set up an ANP network for our valuation problem, and the according information collected so far,
- and discussed the application to our running example.

Now, we have the values for our requirements in place, but still it is not clear which requirements to select even for our small example.

Chapter 20

Optimization of Requirements

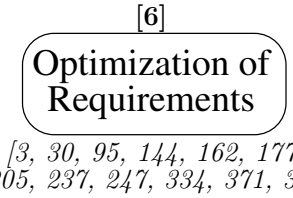


Figure 20.1.: *Optimization of Requirements*

In this step, all the requirements, their relations and the corresponding values are prepared in such a way that they can form the input to an optimization model. We describe the setup of the optimization model (Section 20.2). Using the optimization model, it is possible to compute the optimal set of requirements regarding the optimization goals and the valuation of the requirements automatically. For our optimization model, we decided to use a *weighted sum criterion* method which is one of the most commonly used optimization methods in engineering (Marler and Arora, 2004 [251]). We discuss this decision in Section 20.1. We apply the optimization to our running example in Section 20.3[1]. In Section 20.4[2], we discuss related work, and then conclude this chapter in Section 20.5[3]. This chapter is based on Alebrahim, Choppy, Faßbender, and Heisel (2014 [6])[4].

20.1. Selecting an Optimization Method

We base our decision on a work of Marler and Arora (2004 [251]), who surveyed different optimization methods and compared them with regards to different properties. For our solution we were looking for a method which

- easily integrates with our overall process in general and the ANP in particular,
- only has to be formulated once and is then re-usable in general without the need to adapt it for the system-to-be at hand,
- allows tool support,
- does not require the end user to have any special expertise,
- computes a solution which is really optimal, and
- has a reasonable computational complexity[5].

The *weighted sum criterion* method is a good choice with respect to these criteria, because

[1] Page 348
[2] Page 349
[3] Page 350
[4] The optimization method and model is a contribution of the author.
[5] Computing a solution within minutes is acceptable in our case, but a method taking days for even small systems would be beyond reasonable bounds.

- the values generated by the ANP, and all the elements, such as goals, requirements, and so forth, can be easily reflected in the optimization model,
- once the optimization model is formulated, it is re-usable in general as long as all the information required is available, and it is even possible to make the optimization model modular to skip missing information,
- there are many tools for solving such optimization models,
- the end user does not need to know anything about the optimization itself,
- the found solution is an optimal solution, and
- a solution is found within a reasonable time[6].

Some strong contenders considered in the beginning were physical programming and genetic programming[7]. They both have the advantage that they provide a set of solutions with the same "value". Hence, one can choose freely from this set and gets a solution which is as good as the other ones. This is very convenient and useful for the end users, because they get aware of different "configurations" of the system-to-be, which all satisfy the different stakeholders to the same extent. Using the weighted sum criterion method, one gets only one solution.

But there are also serious drawbacks for physical programming and genetic programming. For physical programming, the optimization model needs to be adapted to each system-to-be it is used for, and the user needs some expertise in the field of physical programming. From our point of view, this is not acceptable. For genetic programming, it turned out that it only finds a set of solutions with equal value, but there is no proof that these solutions are from the pareto-optimal front. For an optimal solution it is neither sufficient nor even necessary to be found by a genetic algorithm. Hence, after running a genetic algorithm one has some solutions which might not be optimal, which is not acceptable.

20.2. The Optimization Model

In this section we describe our optimization model, including the parameters (Section 20.2.1), decision variables (Section 20.2.2), the target function (Section 20.2.3), and the constraints (Section 20.2.4[8]).

20.2.1. Parameters

The parameters of the optimization model are described in the following. The parameters contain sets which describe the entities of the different models as elicited in the previous activities of the overall process (Section 20.2.1.1). Additionally, there are parameters describing the relations between the entities, and the values for the entities (Section 20.2.1.2).

20.2.1.1. Sets

The entities of importance are the goals (hard goals, soft goals, tasks) as described by the goal model (see Section 11.1[9]), the processes and contained activities (functions) (see Section 11.2[10]),

[6] For the running example in less than a second

[7] We considered all of the methods enumerated in (Marler and Arora, 2004 [251]), but for reasons of brevity we only discuss the important ones

[8] Page 346

[9] Page 203

[10] Page 206

and the requirements for our system-to-be including functional (see Section 12.2[11]) as well as quality requirements (see Chapter 13[12], and Chapter 16[13]), and the according alternatives (see Section 18.3[14]).

$$\mathsf{G} = \{G1, G2, \ldots, Gz\} \quad \text{Set of goals} \tag{20.1}$$
$$\mathsf{P} = \{P1, P2, \ldots, Py\} \quad \text{Set of processes} \tag{20.2}$$
$$\mathsf{A} = \{A1, A2, \ldots, Ax\} \quad \text{Set of activities} \tag{20.3}$$
$$\mathsf{FR} = \{R1, R2, \ldots, Rw\} \quad \text{Set of functional requirements} \tag{20.4}$$
$$\mathsf{QR} = \{Q1, Q2, \ldots, Qv\} \quad \text{Set of quality requirements} \tag{20.5}$$
$$\mathsf{R} = \mathsf{FR} \cup \mathsf{QR} \quad \text{Set of requirements} \tag{20.6}$$

20.2.1.2. Relations / Value Coefficients

Of course, we also need to relate the goals, processes, activities, and requirements, as well as we have to codify information such as *must have* or the particular value.

$$G^{must}_{i \in \mathsf{G}} \in \{0,1\} \quad \text{Determines whether a goal } i \text{ has to be in the solution (1) or not (0)} \tag{20.7}$$
$$G^{root}_{i \in \mathsf{G}} \in \{0,1\} \quad \text{Determines whether a goal is one of the root goals (has no parent) (1) or not (0)} \tag{20.8}$$
$$P^{must}_{m \in \mathsf{P}} \in \{0,1\} \quad \text{Determines whether a process } m \text{ has to be in the solution (1) or not (0)} \tag{20.9}$$
$$R^{initial}_{k \in \mathsf{R}} \in \{0,1\} \quad \text{Determines whether a requirement } k \text{ is part of the initial set of requirements (1) or not (0)} \tag{20.10}$$
$$R^{must}_{k \in \mathsf{R}} \in \{0,1\} \quad \text{Determines whether a requirement } k \text{ has to be in the solution (1) or not (0)} \tag{20.11}$$
$$R^{value}_{k \in \mathsf{R}} \in \mathbb{R} \quad \text{Determines the value of a requirement } k \tag{20.12}$$
$$G2G^{and/or/xor}_{i,j \in \mathsf{G}} \in \{0,1\} \quad \text{Determines whether a goal } j \text{ is sub-goal to goal } i \text{ in an (AND/OR/XOR)-relation (1) or not (0)} \tag{20.13}$$
$$G2G^{deny}_{i,j \in \mathsf{G}} \in \{0,1\} \quad \text{Determines whether a goal } j \text{ denies goal } i \text{ (1) or not (0)} \tag{20.14}$$
$$G2R^{and/or/xor}_{i \in \mathsf{G}, k \in \mathsf{R}} \in \{0,1\} \quad \text{Determines whether a requirement } k \text{ is required to fulfill goal } i \text{ in an (AND/OR/XOR)-relation (1) or not (0)} \tag{20.15}$$
$$P2A_{m \in \mathsf{P}, n \in \mathsf{A}} \in \{0,1\} \quad \text{Determines whether an activity } n \text{ is part to the process } m \text{ (1) or not (0)} \tag{20.16}$$
$$A2R_{n \in \mathsf{A}, k \in \mathsf{R}} \in \{0,1\} \quad \text{Determines whether a requirement } k \text{ is required to execute activity } n \text{ (1) or not (0)} \tag{20.17}$$
$$R2R^{deny}_{k,l \in \mathsf{R}} \in \{0,1\} \quad \text{Determines whether a requirement } l \text{ denies requirement } k \text{ (1) or not (0)} \tag{20.18}$$

[11] Page 214
[12] Page 221
[13] Page 281
[14] Page 327

$R2R^{requires}_{k,l \in \mathsf{R}} \in \{0,1\}$ Determines whether a requirement l requires requirement k (1) or not (0) (20.19)

$R2R^{complement}_{k \in \mathsf{FR}, l \in \mathsf{QR}} \in \{0,1\}$ Determines whether quality requirement l complements functional requirement k (1) or not (0) (20.20)

$R2R^{alternative}_{k,l \in \mathsf{R}} \in \{0,1\}$ Determines whether requirement l is an alternative for requirement k (1) or not (0) (20.21)

The inputs to this optimization model are the goal model with interactions (see Figure 11.2[15]), the processes and activities (see Section 11.2[16], and the requirements with interactions, alternatives (see Table 18.1[17] and Table 18.4[18], Table 18.5, and Table 18.6[19]), and their values (see Chapter 19[20]). The information about AND, OR, XOR - relations between goals is added to the corresponding coefficients $G2G^{and/or/xor}$. For the goal interactions we only add the information about goals denying each other, using the coefficient $G2G^{deny}$. The other positive or negative interactions are already considered when valuating a requirement using ANP. The processes and according activities are related using $P2A$. For the requirements, we capture the information about denying requirements in $R2R^{deny}$, the information about complementing requirements in $R2R^{complement}$, and whether a requirement is an alternative for another requirement in $R2R^{alternative}$. If a goal, process, or requirement has to be in the solution, is expressed using G^{must} , P^{must}, or R^{must}, respectively. We also codify the information if a goal is a root goal which has no parent goal G^{root}. For the requirements we also model the information which value the requirement has (R^{value}), and if a requirement was already in the initial set of requirements or not using $R^{initial}$. Last, we have to relate goals and requirements using $G2R^{and/or/xor}$, and activities and requirements using $A2R$.

20.2.2. Decision variables

The decision variables describe our solution in the end. For each entity as described by our parameter sets (Section 20.2.1.1), it has to be decided whether this entity is in the solution or not.

$g_{i \in \mathsf{G}} \in \{0,1\}$ Determines whether goal i is part of the solution (1) or not (0) (20.22)

$p_{m \in \mathsf{P}} \in \{0,1\}$ Determines whether process m is part of the solution (1) or not (0) (20.23)

$a_{n \in \mathsf{A}} \in \{0,1\}$ Determines whether activity n is part of the solution (1) or not (0) (20.24)

$r_{k \in \mathsf{R}} \in \{0,1\}$ Determines whether requirement k is part of the solution (1) or not (0) (20.25)

The solution is given with respect to the the decision variables g_i, p_i, a_i and r_k, which indicate whether a goal, process, activity or requirement is in the solution or not.

[15] Page 204
[16] Page 206
[17] Page 322
[18] Page 326
[19] Page 329
[20] Page 333

20.2.3. Target function

The target of our optimization is to minimize the difference between the initial set of requirements, which contains some unresolved conflicts but is ideal in the sense that all goals and requirements of each stakeholder are completely covered, and the compromise set of requirements, which contains no conflicts but relaxed original requirements or even excludes some goals and requirements.

$$Minimize((\sum_{k \in \mathsf{R}} R_k^{initial} * R_k^{value}) - (\sum_{k \in \mathsf{R}} r_k * R_k^{value})) \quad \text{Minimize difference between ideal solution with conflicts and compromise solution} \tag{20.26}$$

Note that the target function does not look like the regular form for multi-objective optimization (see Section 2.2.6[21]). Indeed, the target function only optimizes the values of the requirements. The reason is that we moved the aggregation of different objectives and stakeholders to ANP. Hence, the value for a requirement already reflects the views of different stakeholders on different dimensions, such as benefits and costs. As a result, we do a hidden multi-objective optimization, regarding the target function. This simplifies the optimization, but a more fine-grained and detailed optimization model which integrates the ANP computations might produce even better results, with the drawback that the valuation cannot be executed using another valuation method. Note that this does not invalidate the claim that the optimization model does find an optimum with regards to its inputs. But the optimality is regarding the model which is optimized, and a model is always a simplification of the real world. If it is possible to increase the fit with reality, and if this increased fit has an impact on the found solution is a topic for future research.

20.2.4. Constraints

As the solution has to assure several properties, there are some constraints. For example, it is not allowed to have two requirements in the solution which deny each other. Another example is the property that a goal is only allowed in the solution if at least one parent is in the solution and it is fulfilled with respect to related sub-goals or requirements. All constraints are formulated in the following.

$$\overbrace{G_i^{must} - g_i \leq 0}^{\text{Assures that goal } i \text{ is in the solution whenever it is a must goal}} \quad \forall i \in \mathsf{G} \tag{20.27}$$

$$\overbrace{\underbrace{(1 - G_i^{root})}_{\text{is 0 in case } i \text{ is an root goal}} * (g_i - \sum_{j \in \mathsf{G}} \underbrace{(g_i * (G2G_{j,i}^{and} + G2G_{j,i}^{or} + G2G_{j,i}^{xor}))}_{\text{1 if } i \text{ is in the solution and } j \text{ is its child.}} \leq 0}^{g_i \text{ has to be 0 if no parent goal is in the solution as long as it is no root goal}} \quad \forall i \in \mathsf{G} \tag{20.28}$$

$$\overbrace{(1 - \underbrace{\prod_{j \in \mathsf{G}} (1 - \overbrace{(g_j * G2G_{i,j}^{deny})}^{\text{1 if } j \text{ denies } i \text{ and } j \text{ is in the solution}})}_{\text{1 if no goal denying } i \text{ is in solution}}) * g_i = 0}^{g_i \text{ has to be 0 if any other goal denying it is in the solution. Otherwise free choice.}} \quad \forall i \in \mathsf{G} \tag{20.29}$$

[21] Page 39

$$\overbrace{(1-\underbrace{\prod_{j\in \mathsf{G}}\overbrace{((1-G2G^{and}_{i,j})+(G2G^{and}_{i,j}*g_j))}^{\text{1 if } j \text{ is not an AND sub-goal of } i \text{ or } j \text{ is an AND sub-goal and } j \text{ is in the solution}}}_{\text{1 if all AND sub-goals are in the solution}})*g_i}^{g_i \text{ has to be 0 if an AND sub-goal is not in the solution. Otherwise free choice.}}=0 \quad \forall i \in \mathsf{G} \tag{20.30}$$

$$\underbrace{(1-\overbrace{\prod_{j\in \mathsf{G}}(1-G2G^{or}_{i,j})}^{\text{0 if } i \text{ has at least one OR sub-goal}})*(g_i-\overbrace{\sum_{j\in \mathsf{G}}(G2G^{or}_{i,j}*g_j)}^{\text{Sum of OR sub-goals in the solution}})}_{g_i \text{ is free to choose when at least 1 OR sub-goal of } i \text{ is in the solution, otherwise 0.}}\leq 0 \quad \forall i \in \mathsf{G} \tag{20.31}$$

$$\underbrace{(1-\overbrace{\prod_{j\in \mathsf{G}}(1-G2G^{xor}_{i,j})}^{\text{0 if } i \text{ has at least one XOR sub-goal}})*(g_i*(1-\overbrace{\sum_{j\in \mathsf{G}}(G2G^{xor}_{i,j}*g_j)}^{\text{Sum of XOR sub-goals in the solution}}))}_{g_i \text{ is free to choose when exactly 1 XOR sub-goal of } i \text{ is in the solution, otherwise 0.}}=0 \quad \forall i \in \mathsf{G} \tag{20.32}$$

$$\overbrace{P^{must}_m-p_m\leq 0}^{\text{Assures that process } m \text{ is in the solution whenever it is a must process}} \quad \forall i \in \mathsf{G} \tag{20.33}$$

$$\overbrace{(1-\underbrace{\prod_{n\in \mathsf{A}}\overbrace{((1-P2A_{m,n})+(P2A_{m,n}*a_n))}^{\text{1 if } n \text{ is not an activity of } m \text{ or } n \text{ is an activity of } m \text{ and } n \text{ is in the solution}}}_{\text{1 if all activities related to } m \text{ are in the solution}})*p_n}^{p_n \text{ has to be 0 if an activity related to } m \text{ is not in the solution. Otherwise free choice.}}=0 \quad \forall m \in \mathsf{P} \tag{20.34}$$

$$\overbrace{(1-\underbrace{\prod_{k\in \mathsf{FR}}\overbrace{((1-A2R_{n,k})+(A2R_{n,k}*r_k))}^{\text{1 if } k \text{ is not an requirement related } n \text{ or } k \text{ related to } n \text{ and } k \text{ is in the solution}}}_{\text{1 if all functional requirements related to } n \text{ are in the solution}})*a_n}^{a_n \text{ has to be 0 if a functional requirement related to } n \text{ is not in the solution. Otherwise free choice.}}=0 \quad \forall n \in \mathsf{A} \tag{20.35}$$

$$\overbrace{(1-\underbrace{\prod_{k\in \mathsf{R}}\overbrace{((1-G2R^{and}_{i,k})+(G2R^{and}_{i,k}*r_k))}^{\text{1 if } k \text{ is not an AND requirement of } i \text{ or } k \text{ is an AND requirement and in the solution}}}_{\text{1 if all AND requirements of } i \text{ are in the solution}})*g_i}^{g_i \text{ has to be 0 if an AND requirement is not in the solution. Otherwise free choice.}}=0 \quad \forall i \in \mathsf{G} \tag{20.36}$$

$$\underbrace{(1-\overbrace{\prod_{k\in \mathsf{R}}(1-G2R^{or}_{i,k})}^{\text{0 if } i \text{ has at least one OR requirement}})*(g_i-\overbrace{\sum_{k\in \mathsf{R}}(G2R^{or}_{i,k}*r_k)}^{\text{Sum of OR requirements in the solution}})}_{g_i \text{ is free to choose when at least 1 OR requirement of } i \text{ is in the solution, otherwise 0.}}\leq 0 \quad \forall i \in \mathsf{G} \tag{20.37}$$

$$\underbrace{\overbrace{(1-\prod_{k\in \mathsf{R}}(1-G2R^{xor}_{i,k}))}^{\text{1 if } i \text{ has at least one XOR requirement}}*(g_i*\overbrace{(1-\sum_{k\in \mathsf{R}}(G2R^{xor}_{i,k}*r_k)}^{\text{0 if one XOR requirement is in the solution}})}_{g_i \text{ is free to choose when exactly 1 XOR requirement of } i \text{ is in the solution, otherwise 0.}}\leq 0 \quad \forall i \in \mathsf{G} \tag{20.38}$$

$$\overbrace{R_k^{must} - r_l \leq 0}^{\text{Assures that a requirement } k \text{ is in the solution, when it is a must requirement}} \qquad \forall k \in \mathsf{R} \tag{20.39}$$

$$\overbrace{(1 - \underbrace{\prod^{l \in \mathsf{R}} (1 - \overbrace{(r_l * R2R_{k,l}^{deny})}^{\text{1 if } l \text{ denies } k \text{ and } l \text{ is in the solution}})}_{\text{1 if no requirement denying } k \text{ is in solution}}) * r_k = 0}^{r_k \text{ has to be 0 if any other requirement denying it is in the solution. Otherwise free choice.}} \qquad \forall k \in \mathsf{R} \tag{20.40}$$

$$\overbrace{(1 - \underbrace{\prod^{l \in \mathsf{R}} (1 - \overbrace{(r_l * R2R_{k,l}^{alternative})}^{\text{1 if } l \text{ is an alternative for } k \text{ and } l \text{ is in the solution}})}_{\text{1 if no requirement, which is an alternative for } k, \text{ is in solution}}) * r_k = 0}^{r_k \text{ has to be 0 if any other alternative requirement is in the solution. Otherwise free choice.}} \qquad \forall k \in \mathsf{R} \tag{20.41}$$

$$\overbrace{(1 - \underbrace{\prod^{l \in \mathsf{R}} \overbrace{((1 - R2R_{k,l}^{requires}) + (R2R_{k,l}^{requires} * r_k))}^{\text{1 if } l \text{ is not an required requirement of } k \text{ or } l \text{ is an required requirement and in the solution}}}_{\text{1 if all required requirements of } k \text{ are in the solution}}) * g_i = 0}^{r_k \text{ has to be 0 if an required requirement is not in the solution. Otherwise free choice.}} \quad \forall k \in \mathsf{R} \tag{20.42}$$

$$\underbrace{\overbrace{(1 - \prod^{l \in \mathsf{R}} (1 - R2R_{l,k}^{complements}))}^{\text{0 if } k \text{ complements no other requirement}} * (r_k - \overbrace{\sum_{l \in \mathsf{R}} (R2R_{l,k}^{complements} * r_l)}^{\text{Sum of complement requirements in the solution}}) \leq 0}_{r_k \text{ has to be 0 if no requirements it complements is in the solution. Otherwise free choice}} \quad \forall k \in \mathsf{R} \tag{20.43}$$

$$\overbrace{r_k - \sum_{i \in \mathsf{G}} \underbrace{(g_i * (G2R_{i,k}^{and} + G2R_{i,k}^{or} + G2R_{i,k}^{xor}))}_{\text{1 if } i \text{ is in the solution and } k \text{ is related to it.}} \leq 0}^{r_k \text{ has to be 0 if no goal , it is related to, is in the solution}} \quad \forall k \in \mathsf{R} \tag{20.44}$$

20.3. Application

For the requirements of our running example the results of applying our optimization solution are as follows. We used LPSolve[22] with the Zimpl (Koch, 2004 [221])[23] plugin for solving the optimization. The optimization model formulated as Zimpl model is shown in Appendix F.1.8[24]. The solution selected by the optimizer covers all goals of the stakeholders. There is only one exception. The collection of information about the customer for selling this information is not covered. Hence, the benefits of selling such information does not outweigh the possible drawbacks. For the content aggregator selling the customer information is not a central concern. Hence, on the goal level the solution is fine.

All functional requirements (R3, R15, R16, R17, and R18) are part of the solution, which is not surprising as the content payment process is a must have process and so are the related functional requirements. Regarding the security requirements for R3, only the integrity requirement R3B

[22] http://lpsolve.sourceforge.net/5.5/
[23] http://zimpl.zib.de/
[24] Page 543

is in the solution in its original form. For confidentiality and availability, R3I and R3M have been selected. Hence, these two security requirements were relaxed for cost reasons. But also the running cost requirement was relaxed as R3M was selected instead of R3G. Here, we have a somehow "meet in the middle" solution. All other cost related requirements were selected as they originally were.

The result already shows that naively picking the requirements according to their ranks will not give the same result but another set of requirements, which is inferior. For example, picking by the ranks would have included R3A and R3C, but no running costs requirement for R3 which is not acceptable for the content aggregator. For such a small example such a situation might be easily noticeable and resolvable by hand, but with an increasing number of requirements this cannot be done that easily anymore. Considering a goal tree which is more complicated, in terms of relations between goals of different stakeholders, the situation gets even more complicated. Whenever a goal is not satisfied, all of its sub-goals are also removed as long as they have no other parent. This also leads to a removal of the related requirements. Managing all of these goal and requirement interactions is hardly possible for a human for bigger scenarios. But using the optimization model, all balancing and managing of interactions is achieved automatically.

20.4. Related Work

Akker et al. (2005 [3]) present a solution for selecting requirements to be included into the next release. They use a weighted sum method to determine the set of requirements which is optimal regarding a given benefit (single value) for each requirement and the costs to implement it. The only constraint to be met is a given budget which can be spent on the next release. In contrast to our solution, the optimization model is quite simple, because we consider much more information than Akker et al. (2005 [3]) do. For example, they do not consider goals or even relations between requirements. A similar solution is presented by Jung (1998 [205]).

A likewise simple solution, regarding the information considered, is presented by Baker et al. (2006 [30]) for selecting components of the shelf regarding a given budget. Moreover, they propose a greedy algorithm or a simulated annealing (kind of genetic programming) as solution, which both do not guarantee the optimality of the solution. Of course, they outperform other optimization methods regarding the computational complexity, but the optimization problems in requirements selection are far from being that complex (from the point of view of the optimization community) that they cannot be handled by optimization methods which guarantee optimality. There are some other solutions in the filed of release planning which take into account less information than our solution, and which rely on genetic programming such as Greer and Ruhe (2004 [162]), and Trummer et al. (2014 [371]).

Another solution which focuses on component selection considering benefits and costs is proposed by Saliu and Ruhe (2007 [334]). Here the specifically interesting point is that Saliu and Ruhe (2007 [334]) formulate one objective function for benefits and one for costs each. Then they balance those functions making trade-offs which results in a set of solutions which is optimal with regards to a particular trade off done. Hence, the decision maker has to compare the different trade offs and make his / her final decision. Of course, our method considers much more information than the one of Saliu and Ruhe (2007 [334]), and is focused on requirements selection and not component selection, but the idea of making trade offs visible and propose a solution for each kind of trade off might be worth a look in the future.

Finkelstein et al. (2009 [144]) present a requirements selection method which takes into account different kinds of customers, a set of requirements, the perceived value for each requirement by each customer, and costs for implementing each requirement. The constraint to be met is the overall budget, while the optimization function not only tries to maximize the overall value, but also takes into account so called "fairness" functions to avoid that one or more customers are

completely neglected when choosing a solution. The method by Finkelstein et al. (2009 [144]) tries to achieve the same goal of selecting the optimal system regarding the perceived benefits by different stakeholders. In consequence, there are some similarities, but the method takes into account less information, for example no relations between requirements are considered, and the optimization is executed using a genetic programming method.

Letier et al. (2014 [237]) present a method which lets one select an optimal set of requirements taking into consideration a benefit, a cost, and a risk value for each requirement. The solution selection is done using multi-objective optimization methods[25]. The central concern addressed by the overall solution is uncertainty. Letier et al. (2014 [237]) do not assume that each value is 100% sure, but that the actual value can deviate. In consequence, a desired solution should be robust against changes in the values. Letier et al. (2014 [237]) use a Monte Carlo simulation to generate different scenarios from the different values and the according possible deviations to find a robust solution. While the optimization model used by Letier et al. (2014 [237]) is reflecting only some of the information we consider, the uncertainty aspect is truly a point for future work.

van den Akker et al. (2008 [376]) present a method which uses the weighted sum method to formulate an optimization model. This model reflects the requirements and the relations between the requirements for system to be. The optimization then tries to find an optimal solution regarding a given effort, time and cost budget. While van den Akker et al. (2008 [376]) do not cover all the information regarding goals, stakeholders, and so forth, with regards to the relation between requirements they allow more detail. van den Akker et al. (2008 [376]) model all the relations we use, but even some more, such as cost synergies, which means the costs for implementing one requirement decreases if another requirement is also implemented because of, for example, reuse, or synergies in benefits, which means that the benefit of having two requirements is greater than just the sum of the benefits of the two requirements. Additionally, van den Akker et al. (2008 [376]) model different actions one can take to meet, for example, a certain deadline, such as involving another development team or externals, or extending the deadline. For all such actions, penalties are introduced. It might be worthwhile to have a look at the method of van den Akker et al. (2008 [376]) to extend our method on the requirements level. Still, we doubt that the information needed for, for example, quantify synergies can be collected that easily.

20.5. Conclusion

In this chapter, we have shown how all the information collected so far is used in the end to conduct an informed requirements selection using optimization methods. In particular, we have discussed

- which optimization method we have chosen for our method (weighted sum criterion) and why,
- how the optimization model looks like and how the different inputs are reflected,
- the application of the optimization model to the running example, and
- other method on the same topic of requirements selection using optimization.

[25]The authors do not get more specific than that.

Part V.

Discussion

Chapter 21

Tool Support

The tool support for the process proposed in this thesis relies on different state of the art technologies. The tool is a model-based tool realized using the eclipse platform[1]. As eclipse is implemented in Java, which can be used on many different operating systems (OS), also the tool is not restricted to any OS. Figure 21.1 gives an overview of the technologies used, which we discuss in Section 21.1, the architecture, which we discuss in Section 21.2, and the actual plugins developed, which we discuss in Section 21.3. Section 21.4[2] concludes this chapter.

21.1. Used Technologies

In the following, we discuss the technologies we used to realize our tool.

Eclipse Eclipse is a rich client platform (RCP) which allows the development of general purpose tools. Eclipse consists of different components which allow to, for example, develop graphical user interfaces using the standard widget toolkit (SWT[3]), develop MVC driven tools (JFace), and so forth. In Eclipse, one can use bundles and the OSGi life cycle as defined by the OSGi[4] to enable the modularization of a tool. The Eclipse IDE is an IDE implemented using the Eclipse RCP, which provides a extensive ecosystem of existing technologies which are realized as plugins for the Eclipse IDE. Hence, within the Eclipse IDE

[1] www.eclipse.org
[2] Page 357
[3] https://www.eclipse.org/swt/
[4] http://www.osgi.org/Main/HomePage)

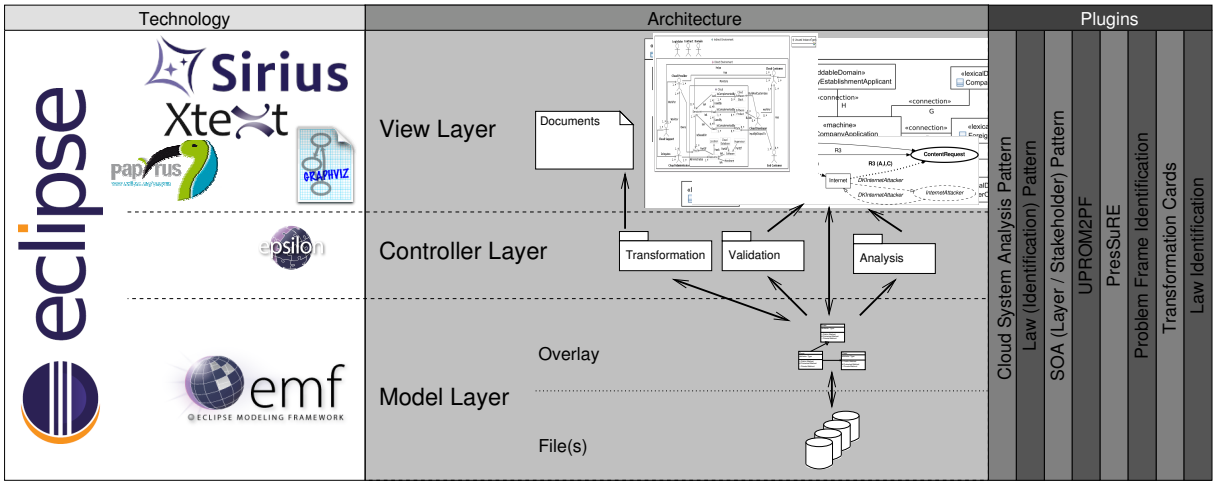


Figure 21.1.: *Architecture of the Tool*

it is easy to reuse existing technologies to develop own plugins which then also integrate in the Eclipse IDE, which allows the user to have one tool integrating all the functionality he / she needs.

Epsilon Epsilon is a family of languages and tools for code generation, model-to-model transformation, model validation, comparison, migration and refactoring that work out of the box with EMF and other types of models[5]. The different languages are briefly explained in the following:

EOL The epsilon object language (EOL) is an imperative programming language for creating, querying and modifying EMF models. It combines the idea of imperative programming languages such as Javascript and and the object constrain language (OCL) for querying models. As such, it provides all the usual imperative features such as statement sequencing, variables, for and while loops, if branches, and so forth, and all the features of OCL as OCL is completely embedded into EOL[6].

ETL "ETL [epsilon transformation language] is a hybrid, rule-based model-to-model transformation language built on top of EOL. ETL provides all the standard features of a transformation language but also provides enhanced flexibility as it can transform many input to many output models,and can query/navigate/modify both source and target models."[7]

EVL "EVL [epsilon validation language] is a validation language built on top of EOL. In their simplest form, EVL constraints are quite similar to OCL constraints. However, EVL also supports dependencies between constraints (e.g. if constraint A fails, don't evaluate constraint B), customizable error messages to be displayed to the user and specification of fixes (in EOL) which users can invoke to repair inconsistencies. Also, as EVL builds on EOL, it can evaluate inter-model constraints (unlike OCL)". [8]

EGL "EGL [epsilon generation language] is a template-based model-to-text language for generating code, documentation and other textual artefacts from models. EGL supports content-destination decoupling, protected regions for mixing generated with hand-written code, and template coordination."[9]

EWL "EWL [epsilon wizard language] is a language tailored to interactive in-place model transformations on user-selected model elements (unlike ETL which operates in a batch mode). EWL is particularly useful for automating recurring model editing tasks (e.g. refactoring, applying patterns or constructing subtrees consisting of similar elements). EWL is integrated with EMF/GMF and as such, wizards can be executed from within EMF and GMF editors."[10]

ECL "ECL [epsilon comparison language] is a hybrid, rule-based language for comparing homogeneous or heterogeneous models. ECL can be used to establish the correspondences on which models can be merged using the merging language of Epsilon, or for transformation testing."[11]

EPL While ECL is sufficient to compare certain model elements of different models, EPL [epsilon pattern language] is used for detecting larger constructs, for example, several elements which are related to each other, within one model.

Sirius "Sirius is an Eclipse project which allows you to easily create your own graphical modeling

[5] http://www.eclipse.org/epsilon/
[6] http://www.eclipse.org/epsilon/doc/eol/
[7] http://www.eclipse.org/epsilon/doc/etl/
[8] http://www.eclipse.org/epsilon/doc/evl/
[9] http://www.eclipse.org/epsilon/doc/egl/
[10] http://www.eclipse.org/epsilon/doc/ewl/
[11] http://www.eclipse.org/epsilon/doc/ecl/

workbench by leveraging the Eclipse Modeling technologies, including EMF and GMF. The modeling workbench created is composed of a set of Eclipse editors (diagrams, tables and trees) which allow the users to create, edit and visualize EMF models."[12]

EMF "The EMF [eclipse modeling framework] project is a modeling framework and code generation facility for building tools and other applications based on a structured data model. From a model specification described in XMI, EMF provides tools and runtime support to produce a set of Java classes for the model, along with a set of adapter classes that enable viewing and command-based editing of the model, and a basic editor."[13]

xText "Xtext is a framework for development of programming languages and domain specific languages."[14] Xtext also allows to define textual editors for EMF models.

Papyrus "Papyrus is aiming at providing an integrated and user-consumable environment for editing any kind of EMF model and particularly supporting UML and related modeling languages such as SysML and MARTE. Papyrus provides diagram editors for EMF-based modeling languages amongst them UML 2 and SysML ....
Papyrus also offers a very advanced support of UML profiles that enables users to define editors for DSLs based on the UML 2 standard."[15]

Graphviz "Graphviz is open source graph visualization software. Graph visualization is a way of representing structural information as diagrams of abstract graphs and networks."[16]

21.2. Architecture

The tool is modularized along the different activities (and sub-activities) within proposed process. This means that for most of the activities a separate eclipse plugin exists. Those plugins can be used in isolation, but they also provide capabilities to integrate all steps. Each plugin itself is further modularized using the model view controller (MVC) design pattern.

The model definition and storage (model layer) is implemented using the eclipse modeling framework. EMF provides capabilities to define meta-models which are then used to generate file structures for storing a meta-model compliant model, an overlay to access such a model programmatically, and tree-editors to create and manipulate a model.

For model generation, validation and analysis (controller layer) several languages of the Epsilon language family are used. They offer capabilities for, for example, the transformation between a process model and an according UML problem frames representation for further use with other plugins within the UML4PF ecosystem, for checking a model for, for example, missing or erroneous information within a model, or information which is inconsistent between models, and for guiding through a semi-automated transformation.

On top of these layers, the Sirius framework is used to implemented graphical editors (view layer) to create and manipulate the models. Xtext is used to provide textual model editing capabilities. The UML4PF tooling uses the Papyrus UML editor and the UML profile mechanism for viewing and editing UML models. Additionally, several types of documents, for example pdfs, are generated for usage by the user. Graphviz is used to vizualize graphs and store those graphs as documents.

[12] https://eclipse.org/sirius/
[13] https://eclipse.org/modeling/emf/
[14] https://eclipse.org/Xtext/
[15] https://eclipse.org/papyrus/
[16] http://www.graphviz.org/

21.3. Plugins

In the following we briefly introduce the different plugins, relate them to the activities they support, and state there maturity of the plugins. We distinguish between the maturity levels of *market ready*, which means that the tool is in a shape it can be actually used in production, *research prototype*, which means the tool is fully functional and usable in general, but still has some rough edges, and *proof of concept*, which means that the general functionality is working as intended, but still important aspects are missing to enable others to work with the tool.

UML4PF The UML4PF tool lets one model the problem frame notation using the UML. It therefore provides different profiles as well as validation capabilities. It makes use of Papyrus and its UML profile mechanisms for modeling purposes. It is the central tool for modeling context diagrams, problem diagrams, domain knowledge diagrams, and so forth, within the different activities which make use of the problem frame notation. UML4PF is a research prototype.

Cloud System Analysis Pattern The Cloud System Analysis Pattern (CSAP) tool was developed in the ClouDAT[17] project and one of the project partners (ITESYS[18]) is currently preparing the usage within future business projects. This tool allows to collect all the information for *context elicitation* in a cloud setting as described in Chapter 8[19], Chapter 9[20], and Chapter 10[21]. The CSAP tool makes use of Sirius for graphical modeling purposes, and Epsilon for transformation, validation, and document generation purposes. The CSAP tool is market ready.

SOA (Layer /Stakeholder) Pattern The SOA (Layer / Stakeholder) Pattern (SOALSP) tool serves the same purpose as the CSAP tool, but for a SOA setting. The SOALSP tool is based on the same technologies like the CSAP tool. In contrast to the CSAP tool, it has only the reached the level of an research prototype.

UPROM2PF The UPROM2PF supports the transformation of UPROM models to context and problem diagrams as described in Chapter 12[22]. Hence, URPOM2PF supports the activities *problem context elicitation*, and *functional requirements elicitation*. It makes use of ETL and EWL for this purpose. It is in an early stage and therefore only a proof of concept.

PresSuRE The PresSuRE tool, which is a research prototype, fully supports the PresSuRE method as described in Chapter 13[23]. Hence, it is related to the activity of *quality requirements elicitation*.

Law (Identification) Pattern The tool for modeling Law (Identification) Pattern (LIP) is based on the same technologies as the CSAP, and the SOALSP tools. Additionally, it provides textual modeling functionality using Xtext. The LIP tool covers the activities *instantiation of law patterns*, *instantiation of law identification patterns (core+context)*, *full instantiation of law identification patterns* (see Figure 14.12[24]) within the *compliance requirements elicitation*. Hence, it is related to Section 15.1[25], Section 16.1[26], and Section 16.2[27]. The tool is still a research prototype.

[17] http://www.cloudat.de/
[18] http://www.itesys.de/
[19] Page 131
[20] Page 151
[21] Page 181
[22] Page 213
[23] Page 221
[24] Page 263
[25] Page 267
[26] Page 281
[27] Page 295

Transformation Cards The transformation card (TC) tool supports the activities described in Section 16.1[28]. It is an implementation of the transformation cards as described in Section 15.2[29]. It makes use of ETL and EWL for transformation purposes. The TC tool is still in its infancy. Hence, it is only a proof of concept.

Problem Frame Identification The problem frame identification (PFI) tool, is related to Section 16.1[30] and used by the TC tool to identify the corresponding problem frame for a problem diagram at hand. It makes use of of EPL to relate problem frames and problem diagrams. It is in the stage of a proof of concept.

Law Identification The law identification tool is an implementation of the algorithm shown in Algorithm 1[31] using EOL. Hence, it covers the activity of *pattern matching* (see Figure 14.12[32]) within the *compliance requirements elicitation*. It is a research prototype.

Beside the tools we developed for supporting our proposed process, we also made use third-party tools for some steps:

UPROM The UPROM tool allows to model FADs, ERDs, and EPCs, and is specifically tailored to support the UPROM method (see Section 2.2.3[33]). Hence, we used for the activity *process elicitation* as described in Section 11.2[34].

Superdecisions "The SuperDecisions software implements the Analytic Network Process for decision making with dependence and feedback developed by Dr. Thomas Saaty."[35]. Obviously, superdecisions is suitable for supporting the activity *valuation of requirements* as described in Chapter 19[36].

LPSolve IDE "The LPSolve IDE (Integrated Development Interface) is a very user friendly ... interface to the lpsolve API. All functionality of lp_solve can be accessed via a graphical and very user friendly application. lp_solve is a Mixed Integer Linear Programming (MILP) solver."[37]. We used LPSolve IDE for conducting the activity *optimization of requirements* as described in Chapter 20[38].

21.4. Conclusion

In this chapter we have briefly introduced the technologies we used for realizing the tool support for different activities within our proposed process. Furthermore, we have outlined the architecture of the different tools and briefly described the different tools.

[28] Page 281
[29] Page 275
[30] Page 281
[31] Page 300
[32] Page 263
[33] Page 33
[34] Page 206
[35] http://www.superdecisions.com/
[36] Page 333
[37] http://lpsolve.sourceforge.net/5.5/index.htm
[38] Page 341

Chapter 22

Conclusion

In this chapter, we conclude the thesis. Therefore, we summarize all the previous chapters in general, and the contributions in particular in Section 22.1. Then, we discuss our research questions and the respective answers in Section 22.2[1]. As MRQ1 is the central research question which gave rise to all the other MRQ, we discuss MRQ1 in detail in Section 22.3[2]. Finally, we outline some future work in Section 22.4[3].

22.1. Summary and Contributions

In this section we give a brief summary of the chapters of this thesis and the important contributions they make. In **Chapter 1**[4] we have introduced the matter of requirements engineering in general and the importance of decision making and requirements selection in particular, taken a closer look at decision making in RE, which information is necessary for an informed requirements selection, which challenges one has to consider when forming a requirements selection method, and outlined a general process to collect the important information and support the decision making. This chapter already contains some contributions:

An overview of information important for an informed requirements selection.

A list of challenges which have to be solved by a method for requirements selection.

Furthermore this chapter already contains the first key contribution:

A process for collecting all necessary information for an informed requirements selection, and for conducting the selection itself.

In **Chapter 2**[5], we have introduced important terms as well as methods and notations used within this work. In the next chapter (**Chapter 3**[6]), we discussed how research in information systems and software engineering should be conducted, and which strategies, processes and methods are available to conduct the research.

The aggregation of different sources on the matter of design science might be seen as small contribution.

[1] Page 363
[2] Page 367
[3] Page 372
[4] Page 3
[5] Page 27
[6] Page 41

Next, we briefly highlighted the methods actually used for conducting research for this thesis (**Chapter 4**[7]). In **Chapter 5**[8], we introduced the running example as well as two cases we have used for validation purposes.

In **Chapter 6**[9], we reviewed and discussed the current state of practice regarding context elicitation in research as well as in the industry. We motivated the importance of context elicitation, collected the problems related to context elicitation, and surveyed the existing solutions. Furthermore, we discussed our idea of context patterns and a pattern language for context elicitation, and elaborated how our work fits in the existing body of knowledge about context elicitation. Contributions of this chapter are:

> We aggregated and discussed different sources of evidence for the importance of context elicitation in requirements engineering.
>
> From a problem & gap study, we derived a list of challenges which have to be mastered by a solution for context elicitation.

In **Chapter 7**[10], we have introduced the meta model and the meta process which are part of the Meta Pattern for describing context patterns. The contributions of this chapter are:

> We defined a meta model for context patterns, and a meta process (method) for deriving context patterns, which help to derive and describe new context patterns.

In the next chapter (**Chapter 8**[11]), we have presented our pattern form, and our catalog of already existing context patterns. The contribution of this chapter is

> A pattern form for context patterns which assists the reader in getting an overview without unnecessary reading, but also provides the information he/she needs when using a specific pattern.

Next, we focused on the relations between context patterns and how to combine context patterns in a meaningful way (**Chapter 9**[12]). The contributions of this chapter are

> A pattern language for context patterns and an according method to find a sequence of context patterns for describing a system-to-be at hand.
>
> Guidance for future context pattern providers who want to integrate their new context patterns into the context pattern language.

Chapter 10[13] served the purpose of exemplifying the application of context patterns using the running example, as well as to present and discuss the validation case for the context patterns. Therefore, we introduced eight validation cases, enumerated the related research questions, the results regarding the questions, and we discussed the results of the validation cases with respect to the requirements for a context elicitation solution, and the general sufficiency of the context patterns for context elicitation.

Overall, the chapters on our context elicitation solution form the second key contribution:

[7] Page 67
[8] Page 75
[9] Page 81
[10] Page 113
[11] Page 131
[12] Page 151
[13] Page 181

Our solution for context elicitation enables a domain-specific context establishment based on patterns for different domains. The context patterns allow a structured elicitation and documentation of relevant stakeholders and technical entities for a system-to-be. Both, the documentation in means of graphical pattern instances and textual template instances as well as the method for collecting the necessary information are explicitly given in each context pattern. Additionally, we also provide the means which are necessary to derive new context patterns and extend our context patterns language.

In **Chapter 11**[14], we showed how to choose and use existing notations and methods for the activities goal elicitation, and process elicitation. Afterward (**Chapter 12**[15]), we motivated our selection of the problem frames notation and method for representing requirements, and how to turn process models into context and problem diagrams. Hence, we contributed

A guided method for creating a context diagram and problem diagrams based on a business process model.

Next (**Chapter 13**[16]), we showed how to break down high level security goals to security requirements which directly complement previously elicited functional requirements. The contribution of this chapter is

A security requirements analysis method which relies on functional requirements only, ensures that crucial security domain knowledge is elicited, and enforces and supports the collaboration of requirement engineers with system stakeholders and security experts.

The next chapters were about the matter of legal compliance requirements engineering. In **Chapter 14**[17], we presented the results of a problem & gap study reviewing the literature in the field of legal compliance requirements engineering. Moreover, we provided insights in the current practice of legal experts, and derived patterns and a method for legal compliance requirements engineering from these insights. Hence, the contributions of this chapter are:

A list of challenges one has to consider when forming a legal compliance requirements solution.

A pattern to model laws (Law Pattern).

A pattern to restructure requirements in way that they are suitable for matching with laws (Law Identification Pattern).

A general process which enables requirement engineers and legal experts to collaborate and supports them in identifying relevant laws, and related requirements for a system-to-be.

The law identification process as presented in Chapter 14[18], contains steps which do not have to be conducted for each execution of the process, because they prepare inputs which can be reused. Hence, these steps are only conducted in case a new input has to be generated or an existing one has to be updated. In **Chapter 15**[19], we described these steps, and contributed:

[14]Page 203
[15]Page 213
[16]Page 221
[17]Page 239
[18]Page 239
[19]Page 267

A detailed guidance for instantiating the Law Pattern for several types of dictates of justice.

Transformation cards for easing the instantiation of Law Identification Patterns, which help to identify the problem class of the requirement at hand, identify the needed domain knowledge for the transformation of problem diagrams to Law Identification Pattern instances, and which contain instructions how to model the domain knowledge as well as instructions for executing the transformation.

In **Chapter 16**[20], we explained the steps which have to be executed for every run of the law identification process. For this purpose we made use of results of an existing requirements engineering (here problem frames) approach for law identification, the aforementioned transformation cards, and the laws modeled using the Law Pattern. In particular, we contributed in this chapter:

A guided method for using the transformation cards.

An algorithm for the matching Law Identification Pattern instances with Law Pattern instances.

A way of preparing the requirements and matching dictates of justice that eases the legal revision by legal experts.

Guidance for deciding which requirements to change and how to change them.

For **Chapter 17**[21], we took a look at the sufficiency of the solution proposed by us for the identification of laws and the subsequent adjustment of requirements, and showed the results of two experimental simulations. Additionally, we discussed whether the method presented takes into consideration the important aspects for a legal compliance requirements engineering method, and we discussed for each step of the method whether there is potential for further improvements and future work.

Overall the chapters on legal compliance requirements engineering form our third key contribution:

A pattern-based and guided method which lets one identify the relevant laws for a system-to-be, which is described in means of functional requirements, and which intertwines the functional requirements with the according legal requirements. This method relies on the collaboration of requirements engineers and legal experts, and bridges the gap between their distinct worlds.

In **Chapter 18**[22] we discussed the matter of interactions between requirements, and outlined methods to detect interactions. Additionally, we presented a method for generating requirements alternatives. In particular, the contribution of this chapter is

A guided method for generating alternatives for quality requirements (in our case security and costs) using so-called requirements relaxation templates.

Then we showed how to use all the information collected so far to set up an ANP model for valuating the requirements (**Chapter 19**[23]). Here, the contribution is

[20] Page 281
[21] Page 307
[22] Page 321
[23] Page 333

A method for forming an ANP which allows to valuate different kinds of requirements with respect to all the important information for making an informed requirements selection.

We finished our proposed overall process by discussing which optimization method we have chosen for our solution (weighted sum criterion) and why, how the optimization model looks like and how the different inputs are reflected, and the application of the optimization model to the running example (**Chapter 20**[24]). Of course, the contribution (which is the fourth key contribution) is

An optimization model which reflects all the important information for an informed requirements selection, which allows then an automatic selection of the optimal set of requirements with respect to the collected information.

Last, we briefly discussed the tool support for our methods in **Chapter 21**[25].

A last contribution, which we consider as fifth key contribution, is that

For each activity we provided a reasoning for the chosen methods, which one can use to find the method for the activity at hand which serves him/her best. Thus, we support the flexibility of the overall process.

22.2. Answers to our Research Questions

The purpose of this work is to answer five main research questions, which we already introduced in Section 1.3[26], and which form the baseline for all the work and results presented in this thesis. In the following we discuss which progress we made throughout this thesis to answer these questions, and what our answers to these questions are:

MRQ2 Which information has to be known for an informed requirements selection?

Answer For the purpose of answering this question, we have conducted a problem & gap study. We surveyed papers on the topic of decision making and requirements selection, and collected statements about the information necessary for making decisions in RE and selecting requirements. The results related to the necessary information of this problem & gap study were presented in Section 1.4.1[27]. Based on these results we answer MRQ2 as follows: *The information about the context of the system-to-be, the relations between the requirements, and the values assigned to each requirement have to be known for a requirements selection. Information about goals, processes, and variants even more improve the selection of requirements.*

MRQ3 How to get the information necessary for requirements selection, and are there best practices available or are there any significant gaps?

Answer The information necessary for conducting an informed requirements selection is of crucial importance for forming a process which enables one to conduct requirements selection, because before actually deciding which requirements to select, the information known about these requirements has to be available. Beside the necessary information, we also derived some important aspects for a requirements selection method from the problem

[24] Page 341
[25] Page 353
[26] Page 8
[27] Page 9

& gap study on decision making in RE and requirements selection. These aspects were discussed in Section 1.4.2[28]. We combined these aspects with the knowledge about the required information, and proposed a general process for collecting all the necessary information and conducting the actual selection of requirements. Figure 1.6[29] shows this process. For each of the activities, we investigated whether there are already existing methods covering the activity. Whenever we found sufficient methods based on a literature review, we selected one of them, and motivated our decision. In case we assumed a gap regarding methods for collecting a particular information, we conducted a more detailed problem & gap study. For the activities of goal elicitation (Section 11.1[30]), process elicitation (Section 11.2[31]), problem context elicitation (Section 12.2[32]), functional requirements elicitation (Section 12.3[33]), quality (in our case security) requirements elicitation (Chapter 13[34]), valuation of requirements (Chapter 19[35]), and optimization of requirements (Chapter 20[36]), we found and used sufficient existing methods. Two small gaps were identified regarding the transition from business process models (in our case UP-ROM models) to functional requirements, and the transition from high level security goals to specific initial security requirements related to the functionality of the system-to-be. For both gaps, we proposed a method (Chapter 12[37], and Chapter 13[38] respectively). Big gaps were identified regarding the context elicitation (see Section 6[39]), and the compliance requirements elicitation for a statute law (the German law in our case) regarding the identification of relevant laws and the concrete relations between dictates of justice and the functionality (see Chapter 14[40]). We closed the identified gaps by proposing a method for context elicitation (Chapter 7[41], Chapter 8[42], Chapter 9[43], and Chapter 10[44]) as well as for legal compliance requirements engineering (Chapter 14[45], Chapter 15[46], Chapter 16[47], and Chapter 17[48]). Two further potential gaps were identified but not investigated in detail for the activities detection of interactions, and generation of alternatives. Even though we did not investigate the gaps in detail, we have proposed some methods for covering these activities as described in Chapter 18[49]. Hence, we can answer MRQ3 as follows: *Figure 1.6[50] shows all the activities necessary for an informed requirements selection. There are two small gaps regarding the transition from business processes to functional requirements, and the transition from security goals to security requirements. Two big gaps are related to context elicitation and compliance requirements elicitation. Two potential big gaps are*

[28] Page 15
[29] Page 19
[30] Page 203
[31] Page 206
[32] Page 214
[33] Page 217
[34] Page 221
[35] Page 333
[36] Page 341
[37] Page 213
[38] Page 221
[39] Page 81
[40] Page 239
[41] Page 113
[42] Page 131
[43] Page 151
[44] Page 181
[45] Page 239
[46] Page 267
[47] Page 281
[48] Page 307
[49] Page 321
[50] Page 19

related to the detection of interactions (especially regarding qualities), and the generation of alternatives. The rest of the activities is covered by best practices. The combination of the methods proposed by us for closing the gaps, and the existing methods selected by us describe how to get the information necessary for requirements selection.

MRQ4 How to elicit all relevant domain knowledge including the stakeholders and the context of the system-to-be in a structured way?

Answer In Chapter 6[51], we discussed the matter of context elicitation, and revealed the gap regarding sufficient solutions. Additionally, we derived challenges from a problem & gap study that a method for context elicitation should cover, introduced the matter of patterns, and discussed the idea of using context patterns for the purpose of context elicitation under the light of the found challenges. Afterward, we introduced a Meta Pattern in Chapter 7[52], which allows one to derive and describe new context patterns, and introduced a catalog of context patterns (Chapter 8[53]) already described by us using a context pattern form. Then, we showed in Chapter 9[54] how to form a pattern language for context elicitation, and how to use the resulting context pattern language to elicit context information for a system-to-be covering different domains. Finally (Chapter 10[55]), we discussed and showed the sufficiency of our proposed context pattern with regards to the challenges a context elicitation method has to cover. The discussion was based on the insights gained from different experiments and so forth. Hence, we can answer MRQ4 as follows: *Our proposed context patterns for context elicitation form a solution for context elicitation which allows one to elicit all relevant domain knowledge including the stakeholders and the context of the system-to-be in a structured way.*

MRQ5 How to address compliance with laws and according qualities such as security and privacy when eliciting the requirements for a system-to-be?

Answer To answer this question, we conducted a problem & gap study in the field of compliance requirements engineering. Two major results were derived from this study. First, we identified a gap regarding legal compliance methods for statute and outcome-based legal systems in general, and for the German law in particular. Additionally, there are hardly any methods regardless of the legal system, which describe how to find the relevant laws for a system-to-be, which allow one to intertwine the already described functionality of a system-to-be with the related dictates of justice, and which integrate the collaboration with legal experts explicitly into the solution. Second, we identified several important aspects a method for compliance requirements engineering has to cover. Our proposed method to close this gap consists of several parts.

First, we analyzed in Chapter 14[56] how legal experts analyze a law at hand with respect to a given case. From the insights gained, we derived and presented a Law Pattern for describing laws on the one hand, and a Law Identification Pattern for restructuring requirements in such a way that they are suitable for a legal analysis on the other hand. Additionally, we introduced a method (Figure 14.12[57]) for using the patterns.

The method itself is consists of activities which can be conducted without a certain system-to-be in mind, and which produce results which are reusable for many different systems-to-

[51] Page 81
[52] Page 113
[53] Page 131
[54] Page 151
[55] Page 181
[56] Page 239
[57] Page 263

be (Chapter 15[58]). The first activity is concerned with instantiating the Law Pattern for a law, and the second activity is concerned with preparing so-called transformation cards, which allow to semi-automatically transform requirements given as problem diagrams into Law Identification Pattern instances. Beside the transformation itself, a transformation card also ensures that missing legal domain knowledge is collected.
Then, we also explained in detail how to use the prepared laws and transformation cards in an actual assessment of a system-to-be for legal compliance. We showed how the Law Identification Pattern instances are created, how these instances are matched with the Law Pattern instances, and how the legal revision, and modification or requirements works (Chapter 16[59]). To show the sufficiency of our proposed method, we discussed the results of an experiment, and related our solution to the aspects a good compliance requirements engineering method should cover.
Additionally, the PresSuRE method as introduced in Chapter 13[60] is suitable to break down vague security goals as given by a law down to concrete security requirements. For example, ProPan (Beckers, Faßbender, Heisel, and Meis, 2012 [42]) works in the same manner for privacy but was not described in detail in this thesis. Hence, we can answer MRQ5 as follows: *Our proposed solution for legal compliance requirements engineering shows how to address compliance with laws and according qualities such as security and privacy when eliciting the requirements for a system-to-be.*

MRQ1 How to select a set of requirements for a system-to-be in a way it satisfies the stakeholders' needs regarding the system-to-be to the maximum (optimal) possible extent?

Answer An important part of the answer to this question is related to the necessary information and how to collect it. This part is already answered. A second important part is that we showed at which point stakeholders are able to influence the decision taken. When following our proposed overall process as shown in Figure 1.6[61], there are many activities which integrate the stakeholders and their view on the system. In the activity goal elicitation (Section 11.1[62]), all stakeholders are able to express the goals they want to achieve when using the system-to-be. In the activity process elicitation (Section 11.2[63]) they are able to provide the processes they want to be supported by the machine. Also for the quality requirements elicitation (Chapter 13[64]), we showed at which point the stakeholders can express further preferences beside the goals which are already taken into account. Moreover, the valuation of requirements (Chapter 19[65]) is specifically tailored to allow the collection of preferences from many different stakeholders. A third important part is that we showed how this information can be actually used for an informed requirements selection. For this purpose, we introduced an optimization model, which makes use of all the collected information in Chapter 20[66]. Hence, we can conclude and answer MRQ1 as follows: *In this whole thesis, we have shown how to collect and relate all the necessary information for an informed requirements selection. The information collection is centered around the expectations of the different stakeholders. Our optimization model takes into account all this information and allows a selection of a set of requirements for a system-to-be in a way it satisfies the stakeholders' needs regarding the system-to-be (as expressed by the collected*

[58] Page 267
[59] Page 281
[60] Page 221
[61] Page 19
[62] Page 203
[63] Page 206
[64] Page 221
[65] Page 333
[66] Page 341

information) to the maximum (optimal) possible extent.

22.3. MRQ1 Revisited

As the answer to MRQ1 is quite brief and was not already discussed in detail in other chapters, we will revisit it in more detail in the following to show that the proposed overall process for requirements selection in general, and the methods selected or proposed for certain activities within the process, form indeed a sufficient process for requirements selection.

For this purpose, we now take a look at the challenges identified in Section 1.4.2[67], and how they are addressed by our method.

Relevant aspects for valuating requirements unknown Within our process, we have a dedicated activity which is concerned with revealing the goals and therefore the important aspects, from a stakeholder perspective, of a system-to-be. Additionally, the goals are related to the requirements in a structured way. Hence, when reaching the ANP it is clear for each stakeholder which requirements have to be compared with respect to which aspects. *We assume this challenge to be thoroughly addressed and reflected throughout the complete overall process.*

Inconsistent judgments The chosen valuation method ANP can handle inconsistent judgments to some extent, and it makes the inconsistencies visible and traceable. Hence, when the inconsistencies are not acceptable, the flawed answers can be revisited[68]. *This challenge is solved by choosing the ANP.*

Absence of quantitative data The ANP makes use of relative, pairwise comparisons, for which one does not need any quantitative data. In case the quantitative data is available, it is possible to use this data instead of comparisons. For example, if the concrete costs are known, the relative value of a requirement regarding costs can be automatically computed. *This challenge is solved by choosing the ANP, too.*

Uncertainty of value This is an aspect which is currently not addressed by the solutions selected for the different activities. The activities of the overall process itself are agnostic to this challenge. Hence, one has to pick only solutions reflecting uncertainty, for example, regarding the goals, interactions, or values. If the uncertainty is not resolved until the optimization activity is reached, also the optimization method has to be adapted to reflect uncertainty. *This challenge is not addressed in detail right now, and of course a point for future work as we will discuss in Section 22.4[69]. Nevertheless, we assume that addressing uncertainty does not change the overall process as such, but only some activities.*

Volatile values Handling changing goals or values is not a problem within our proposed process. When revising the goals, only newly added goals have to be treated, because when removing goals the related requirements, if they do not contribute to other goals as well, are automatically removed by the optimization. Newly added goals might lead to new requirements, or new processes which lead to new requirements. But these new requirements do not trigger a full rerun in the subsequent steps. Within the PresSuRE method only for the new requirements the security domain knowledge has to be additionally elicited and modeled. It is also easy to make those graphs or parts of graphs visible which changed due to the new requirements, and only the changed parts have to be reconsidered. The

[67] Page 15
[68] Note that this step was not exemplified in this thesis as it is already part of the original ANP.
[69] Page 372

same applies to the legal compliance requirements elicitation. The transformation and matching is done for each problem diagram in isolation. Hence, only the new requirements have to be treated, the results of the previous runs remain untouched. Also for the interaction detection, only the comparison between the newly introduced requirements and the already existing ones have to be treated, as the interaction detection methods rely on pairwise comparisons. Again, the alternative generation treats each requirement in isolation. Hence, only the new requirements, in case they cause a conflict, have to be treated. As the ANP also relies on pairwise comparison, the comparisons done for the already existing goals and requirements remain valid. Only for new goals and requirements the comparison has to be conducted. The optimization has to be run once again whenever the input data changes, but this is not an issue as it is fully automatic. To sum up, all the solutions used for conducting activities within our our overall method do not require a full rerun in case something changes, but limit the effort to the newly added goals and requirements. *Hence, this challenge is addressed by our proposed solution.*

Multidimensional values The ANP allows us to aggregate different goals, stakeholders, strategic criteria, relations, and so forth. *Hence, the problem of multidimensional values is handled by the ANP.*

Unknown stakeholders and stakeholder preferences To reveal the stakeholders is one of the goals of the activity context elicitation, and our proposed context patterns enable one to collect the stakeholders as well as their high level preferences for a system to be in a structured way. Within the activity goal elicitation, the preferences are refined. *Hence, this challenge is addressed in a structured way.*

Mandatory requirements The optimization model allows to add must-have goals, processes, and requirements. *Hence, our requirements selection is "must have" aware.*

Kinds of requirements Our overall process has separate activities for eliciting functional as well as quality requirements of different kinds. The interaction detection methods proposed also take into account the differences between functional and quality requirements as well as between different qualities. This is also true for the alternative generation in which different relaxation templates are used for different qualities. Within the valuation, it is clearly described how to model different kinds of requirements and how they are reflected in the valuation. *Hence, our proposed overall process reflects this challenge and makes it explicitly visible, so do our solutions proposed for the different activities. But within this thesis we only investigated the matter of functional, security, compliance, and cost requirements in detail. In consequence, there is the need to extend, for example, the relaxation templates to other qualities.*

Legal requirements The legal compliance topic is addressed by an own activity within the overall process, and the method proposed for this activity allows to identify relevant laws and intertwine them with the requirements of the system-to-be. *This challenge is explicitly treated.*

Number of requirements Using problem frames the overall system-to-be is decomposed based on the requirements, which makes the usage of problem diagrams scalable even for large systems, because the complexity of single problem diagrams is independent of the system size and the number of problem diagrams scales linearly. Additionally, the transformation from business process to problem diagrams lowers the effort of modeling problem diagrams significantly. From our experience, PresSuRE scales well, as most steps rely on the complexity of the environment, because, for example, knowledge about attackers, connection

domains, and so forth is collected, and the environment does not grow significantly when adding further requirements. The same applies for the legal compliance solution, because when using the transformation cards the main effort is related to the collection of legal domain knowledge which is knowledge about the environment. Within the activity detection of interactions, we get a differentiated picture. The method used for interactions between functional requirements is directly tailored to handle large numbers of requirements. It aims at removing all requirements which cannot interact, and only a small set of interaction candidates are left over, which have to be investigated in detail. For the interaction detection between quality requirements, the situation is more complex. Of course there are means to reduce the set of candidates which might interact, but from our experience the number of remaining requirements which might interact remains significant. As those candidates have to be pairwise compared in different scenarios, the number of comparisons grows significantly with each requirement. This might be due to the nature of different qualities and is unavoidable, but from our point of view there is potential for further research regarding scalability. The same applies for the ANP solution we propose. Our proposed way of modeling the ANP already lowers the number of comparisons as it already takes into account the relations between goals and goals, goals and requirements, and requirements and requirements, but still the number of comparisons increases significantly with each requirement. For both, the quality requirements interaction detection and the ANP, the comparisons themselves to do not increase in complexity. Hence, conducting the methods is still easy, but the number of comparisons scale not that well, which leads to a significant effort. The optimization model scales well, because it is fully automatic, and waiting some seconds more is not an issue. *To sum up, the overall process and our chosen solutions scale well in general with an increasing number of requirements, but there are specific points like the interaction detection for quality requirements and the ANP which need to be investigated further to improve their scalability.*

Relations between requirements The matter of relations between requirements is explicitly addressed throughout the overall process and the methods we propose for each of its activities. We discover relations between requirements due to related goals, relations between requirements due to the processes they support, quality requirements are modeled as complements to the functionality of the system-to-be, interactions are explicitly revealed, alternatives are also explicitly modeled, and the optimization model reflects all those relations. *Hence, we consider this challenge to be tackled.*

Relation between goals and requirements The relations between goals and requirements themselves are reflected during the optimization, and for the matter of security we have shown how to break down security goals to initial security requirements.

Alternatives Our overall process addresses the generation of alternatives for requirements explicitly in a separate activity. Additionally, the used goal notation allows to model even goals which are alternatives for each other. The valuation and optimization take both kinds of alternatives explicitly into account. *Hence, the challenge of considering alternatives is thoroughly addressed.*

Volatile requirements See discussion under **volatile values**. *This challenge is addressed by our proposed process and methods.*

Comparable requirements By using the problem frame notation and method, one ensures that the complexity of different requirements does not differ to a large extent, because the overall problem is decomposed into simple problems. From our experience, when using problem frames, most problem diagrams consist of two or three domains beside the machine, and

even four or five domains do not add that much complexity that they are not comparable anymore. *Hence, when using problem frames, this challenge is addressed.*

Decision making as political process As long as all stakeholders accept the overall process and the methods chosen by us for each activity as valid, and in consequence also accept the result of the optimization, the matter of decision making as political process is considered. Many of the activities within the overall process aim at collecting and integrating the different views of different stakeholders and balance them. In case the process and in consequence also the solution for the system-to-be is not accepted, we do not provide any means for negotiations about changing the solution. *Hence, if the overall process and the resulting solution are accepted by the stakeholders, this challenge is treated, but there is no support for negotiation about changing the found solution. It is a topic for future work, if such a support is possible.*

Decision making as collaborative process For each activity in which we do not use existing methods, we provide detailed processes which clearly state who conducts which activity, and we also guide the collaboration by detailed descriptions. *Hence, this challenge is addressed by us.*

Flexibility of the decision process The overall process does not require one to choose the methods for conducting each activity we propose. In the end, one can choose every method which collects and documents the necessary information. Our selection of methods only shows that conducting the overall process is feasible when choosing the appropriate methods for each activity. Of course, we motivate each selected method, and one should take into account the same aspects as we did when choosing a method for an activity. Regarding the flexibility of skipping certain activities the process is only flexible to some extent. All activities related to eliciting the actual requirements are the baseline, because without requirements there is nothing to select. Which qualities one treats depends on the system-to-be, as only the relevant qualities have to be treated. Remember that we also identified must-have, important to have, and nice to have information, beside the requirements, in Section 1.4.1[70]. Hence, from our point of view the context elicitation, relating the requirements, and the valuation of requirements cannot be skipped. In consequence, the optimization model requires at least some high level goals as collected within the context elicitation, the requirements, the relation between requirements, and some kind of value for each requirement. But a detailed refinement of goals can be skipped by just not adding them to the optimization model. The same applies for the processes. If they are not of relevance, the according activity can be skipped, and the corresponding sets in the optimization model are defined as empty sets. In the same way, missing alternatives can be treated. *Hence, our process is fully flexible regarding the methods chosen for each activity, and flexible to some extent regarding conducting certain activities. There are activities which are mandatory as they collect must-have information (context elicitation, functional requirements elicitation, quality requirements elicitation, detection of interactions, valuation of requirements), but other activities can be skipped (goal elicitation, process elicitation, compliance requirements elicitation, generation of alternatives).*

Documentation and aggregation of information Using the agenda principle and notation, we clearly state for each method proposed by us which information is required, which information has to be documented as a result, and which notation / language is used for documenting it. We also describe for each activity within a method how to aggregate the required information and how to produce the results. For those activities for which we

[70] Page 9

use existing methods, we have chosen model-based methods. Hence, it is clear how to document the results. *To sum up, we considered this challenge thoroughly.*

Decision traceability All the information which is used to take a decision in the end is explicitly elicited and documented. Hence, regarding the information used for a decision, our overall process and the methods used for each activity are fully transparent. Additionally, we intertwine the different artifacts, such as processes and functional requirements, soft goals and quality requirements (in our case security and compliance), and so forth. But the reasoning behind specific decisions, for example, the reasoning behind the alternatives generated, or the comparison between two requirements for valuation, is currently not documented. From our point of view, it is not possible to document every reasoning as the effort spent for such a documentation is beyond reasonable bounds, but it might be an interesting topic to identify those decisions which are crucial for the overall selection, and for which a documentation creates a significant benefit. *To sum up, our proposed methods for the different activities of the overall process make the results of decisions explicitly visible and document them. Here, a possible improvement might be the introduction of explicit trace-links. But regarding the reasoning behind decisions taken, our methods fall short. Hence, this is clearly a topic for further research.*

Ease of use Throughout this thesis we have discussed the matter of ease of use for the different methods proposed in isolation. Assuming that the property ease of use is compositional, this should also apply to the overall process, but still we have no evidence, such as an experiment, for this claim. *From our experience, the overall processes and the according methods are easy to use, but still it is a topic for future work to ground this claim in some evidence.*

Preexisting Domain Knowledge Each of the methods proposed by us does not require any preexisting domain knowledge. The relevant knowledge is collected within the methods and the user is guided trough the collection. For example, the context patterns codify general knowledge about a domain in pattern form, and describe in detail how to collect the information specific to the system-to-be at hand. Another example is the legal compliance requirements engineering method in which a requirements engineer is guided to collect legal domain knowledge, or PresSuRE in which in the same manner security domain knowledge is collected. *Hence, when applying our process and methods, one collects all the important and relevant domain knowledge in a structured way. In consequence, there is no required preexisting domain knowledge.*

Tool support *We provide tool support for all of our proposed methods, but still the average maturity of the tools can be improved as well as the integration between the different tools.*

What we see from this discussion about the different challenges and how they are addressed by our solution is that most of them are addressed completely or at least to a large extent. Of course there are aspects which need further improvements and research, but all in all, the challenges are treated. This supports our answer to MRQ1: *In this whole thesis, we have shown how to collect and relate all the necessary information for an informed requirements selection. The information collection is centered around the expectations of the different stakeholders. Our optimization model takes into account all this information and allows a selection of a set of requirements for a system-to-be in a way it satisfies the stakeholders' needs regarding the system-to-be (as expressed by the collected information) to the maximum (optimal) possible extent.*

22.4. Future Work

In this chapter we outline some general future work, which should be conducted to improve the process and methods presented in this thesis even more. As we have already discussed the future work on the matter of context patterns, and our solution for legal compliance requirements engineering in detail in Chapter 10[71] and Chapter 17[72] respectively, we will focus in the following on the other activities and the overall process in general.

More Qualities and Values In this thesis we have focused on security and legal compliance as qualities. Costs were also introduced, but no method given how to elicit such requirements. Hence, a topic for future work is to integrate more qualities and the according methods such as function point analysis (Dreger, 1989 [121]) for costs, or CORAS (Lund et al., 2011 [245]) for risks. For each quality added to the overall process, one needs to have methods to elicit them, to find interactions between this quality and other qualities, to generate alternatives and so forth. Hence, this is a large area for future work.

Possible Gaps in Interaction Analysis and Alternative Generation As already pointed out in Chapter 18[73], the literature review for these activities revealed a possible gap in literature regarding the interaction analysis for quality requirements, as well as for the alternative generation. For this thesis we did not undergo a problem & gap study to investigate these gaps in more detail. Of course, this is a topic for future work.

Integrating ANP and Optimization Model Currently, the ANP is used to compute the values which then are used for the optimization. A value for a requirement computed using the ANP aggregates the different views of different stakeholders, as well as the different values regarding different goals. This simplifies the optimization, but has a drawback. The value computed by ANP reflects *all* goals, but within the optimization some goals might be removed. This might also impact the value of some requirements. Here, we might introduce some error. Hence, it is future work to investigate how serious this impact is. A solution for this problem might be to integrate the ANP computations directly into the optimization model. In consequence, the value of a requirement would change according to the selected goals. If this is feasible, and which effect it has on the computational complexity is up to future work.

More Sophisticated Optimization Currently, the optimization model finds one optimal solution, but in many cases there are more than one solution which form a so-called pareto optimal front. The solutions of a pareto optimal front provide all the same value, but they differ in the trade-offs done to reach a valid solution. It is a topic for future research if there is a way find all those solutions and make the trade-off visible. In consequence, the decision makers would be aware of the options to take, and can select which trade-off suites them best.

Another topic related to the optimization is the matter of uncertainty. Currently, we take all values as certain, but this is a simplification as in reality the odds are low that all values are certain. We try to tackle this problem by using the ANP which allows a more fuzzy reasoning and valuation, but nevertheless it might be an improvement to take uncertainty explicitly into account. There are optimization models which allow one to model uncertainty, and there is also the option to use something like a Monte Carlo simulation in which different scenarios are generated from a set of uncertain values which are then

[71] Page 181
[72] Page 307
[73] Page 321

optimized in isolation and one picks the most "robust" solution regarding all scenarios. Which methods are feasible and produce sufficient results, and what makes a "robust" solution has to be investigated in the future.

Tool Improvement and Integration Currently, the different activities of the overall process are supported by different tools. Some of them are only proofs of concept and should be developed to a usable stage in the future. Moreover, the integration between some of the tools, especially the external tools, is missing. For example, the goal model has to be transferred to the ANP tool by hand, and all the information collected for the optimization has to be codified by hand into the optimization model. The ideal tool suite would have to offer seamless integration between the different models, or, even better, it should be based on one model for all activities.

Traceability Currently, we only model the result of decisions which implicitly also contain the relations between decisions. But we do not support to have explicit trace-links as well as a documentation of the reasoning behind important decisions. It is part of the future work to integrate trace-links as well as investigate for which decision it is important and feasible to document the decision reasoning.

Software Architecture Of course, requirements engineering is not an end in itself, and also the decision making is not restricted to requirements engineering only. As Figure 1.6[74] already indicates, it might be worthwhile to investigate in the future how to support the transition from requirements engineering to the design phase, including the development of a software architecture. Moreover, the Twin Peaks model of Nuseibeh (2001 [289]) indicates that there might even be feedback from the design phase to the requirements phase, which in turn influences the decisions taken.

Effort For some of the methods we use to conduct activities of our overall process, it turned out that they do not scale that well. In particular, the effort taken for interaction detection for qualities and the ANP increase significantly with each requirement added, as both rely on pairwise comparisons. The comparisons itself are easy to take, but the number of comparisons might become a problem. Hence, it is part of the future work how to lower the effort for such "pain points".

Validation While for some of the methods which cover an activity of our overall process we have conducted several experiments, case studies, and so forth (context patterns, PresSuRE, legal compliance requirements engineering), for the rest we only made feasibility studies. This also applies to the overall process. Hence, in the future we need to validate the overall process and some of the methods used for it in more detail to get more insights into their sufficiency and reveal points for improvement.

[74] Page 19

Part VI.

Appendix

Appendix A

Scientific Methodology

In this appendix, we give an more detailed overview of the experiments (Appendix A.1), the case studies (Appendix A.2[1]), and the literature reviews (Appendix A.3[2]) we have conducted in the context of this thesis.

A.1. Experiments

Validation case A (Appendix A.1.1), case E (Appendix A.1.2), case F (Appendix A.1.3) and case G (Appendix A.1.4) were conducted in the form of an experiment.

A.1.1. Validation Case A: Feasibility Experiment

Rationale The topic of this experimental *feasibility study* was to get a first impression whether the Cloud System Analysis Pattern and the Law (Identification) Pattern were usable at all and sufficient for describing a cloud setting and find laws which are of importance for this setting. As these patterns were completely untested, such a test run was necessary to find flaws and gain first insights in their usability and usefulness.

Purpose The experiment should provide insights whether a cloud setting can be described using the Cloud System Analysis Pattern, the information described by the Cloud System Analysis Pattern can be used for further activities, and if the Law (Identification) Pattern work as intended.

The case The case was a banking scenario derived from some real cases we came across when discussing this topic with several companies. In the derived case, a bank wants to migrate paying services to a cloud provider. Several concerns in this context have to be assessed such as costs, security, reliability, and compliance. To judge compliance possibly relevant laws have to be identified. The law used within this case was the federal data protection act.

Research questions

- Is the Cloud System Analysis Pattern suitable to collect all important information in a structured form?
- Is the Cloud System Analysis Pattern convenient to use and the effort spent for using it reasonable?
- Is the collected information suitable for a law identification?

[1] Page 381
[2] Page 387

- Is the combined use of Law Pattern instances and Law Identification Pattern instances sufficient to identify a relevant law?
- Are the Law (Identification) Pattern convenient to use and the effort spent for using them reasonable?

Propositions

- The relationship between initial information, information expected to be collected, and the information actually collected with the patterns was assessed.
- The relationship between the law sections expected to be matched as relevant and the actually matched ones using the Law (Identification) Pattern Instances.

Method of data collection The data collected was of qualitative nature. We conducted several workshop sessions (3 - 5 researchers) in which we used the patterns. Additionally, everyone also kept track of things which worked as well as things which did not work well from his / her point of view. We conducted several iterations of the experiment. We enhanced the pattern after each iteration and reassessed the enhanced pattern in the next iteration.

Methods of data analysis The data analysis was a qualitative, discussion based one. We compared the results with the predefined, expected outcome. Additionally, we discussed our observations we made during the application of the patterns.

Data selection strategy The data was generated by the researchers taking part in this experiment. We had five researchers, who took part in this experiment including the ones who derived the different patterns.

Quality assurance, validity, and reliability As this experiment was only a feasibility study not explicit means were taken to assure the quality. But two of the researchers taking part were experienced post docs, who kept an eye on the experiment itself.

A.1.2. Validation Case E: Experimental Simulation

Rationale The topic of this *experimental simulation* was to assess the Law (Identification) Pattern and the accompanied problem frame integration (transformation cards) using a real case for which the requirements as well as the final judgment of a court is known, and publicly available.

Purpose The experiment should provide information about the precision and recall when using Law (Identification) Pattern instances as well as the effort spent on creating and using the patterns.

The case The case was an actually developed voting system. By its very nature, the field of electronic voting is an interdisciplinary field, where legal and computer scientists work together. During the development of the first voting system used in Germany, this fact was neglected or inadequately considered. Hence, the federal constitutional court of Germany judged in 2009, that using this system for voting in 2005 was unconstitutional. The requirements specification of this voting system is publicly available as Common Criteria Profile for voting systems. So is the final judgment of the federal constitutional court of Germany.

Research questions

- Are the transformation cards sufficient to be integrated into the overall identification process as described by in Section 14.5[3]?
- Does using the transformation cards and the Law (Identification) Patterns lead to an identification of all relevant laws?
- Are the transformation cards and the Law (Identification) Patterns competitive with a pure discussion based assessment involving legal experts?

Propositions

- The relationship between initial information, information expected to be collected, and the information actually collected with the patterns was assessed.
- The relationship between the law sections expected to be matched as relevant and the actually matched ones using the Law (Identification) Pattern Instances.

Method of data collection We selected four laws.

- The BDSG as highly relevant law concerning personal data.
- The BWahlG (Bundeswahlgesetz), which is the law for federal state elections in Germany and also highly relevant.
- The SigG (Signatur Gesetz), a law which regulates the use of digital signatures. This particular law was selected not due to its relevance, but due to the matter that it is somehow related to our case study, e.g. the technological background.
- The PassG (Pass Gesetz), which regulates the use of passports in Germany. This law is clearly irrelevant, even though the passport is a possible authentication means during elections.

The first two laws were also part of the final judgment while the latter two were not mentioned.
We modeled the requirements as formulated by the Common Criteria profile for voting systems using problem frames. In a first run we discussed these requirements with an legal expert and decided which law (sections) should match which requirement and for which reasons they should match.
Then we modeled the four laws using the Law Pattern. And we used the transformation cards to generate the Law Identification Pattern instances. Then we matched the Law Pattern instances and the Law Identification Pattern instances.

Methods of data analysis We assessed the actual matches under the light of our manual matching. Hence, we were looking for missing matches as well as matches we did not expect. Those matches were discussed if we were wrong or the models we were using were missing information which lead to the diverging match. This way we were able to calculate the precision and recall of the method.

Data selection strategy The data was generated by the author as requirements expert, and one legal expert. For some discussions a further researcher of the working group was involved.

Quality assurance, validity, and reliability The main means to assure the quality of the experiment was to do everything only once. Hence, no models were readjusted, opinions changed, and so forth. Additionally, the discussions with the legal expert were completely separated from the application of the method. And for both, the same problem diagrams without any modifications were used.

[3]Page 262

A.1.3. Validation Case F: Feasibility Experiment

Rationale The topic of this experimental *feasibility study* was to get a first impression whether the SOA (Layer / Stakeholder) Pattern were usable at all and sufficient for describing a SOA setting. As these patterns were completely untested, such a test run was necessary to find flaws and gain first insights in their usability and usefulness.

Purpose The experiment should provide insights whether a SOA setting can be described using the SOA (Layer / Stakeholder) Pattern, and if the usage of the patterns was convenient.

The case The case was the same case as used as running example in this work (See Chapter 5[4]). The case was described before any of the context patterns were existing.

Research questions

- Are the SOA (Layer / Stakeholder) Pattern suitable to collect all important information in a structured form?
- Is the use of the SOA (Layer / Stakeholder) Pattern convenient?

Propositions

- The relationship between initial information, information expected to be collected, and the information actually collected with the patterns was assessed.

Method of data collection The data collected was of qualitative nature. We conducted several workshop sessions (3 - 4 researchers) in which we used the patterns. Additionally, everyone also kept track of things which worked as well as things which did not work well from his / her point of view.

Methods of data analysis The data analysis was a qualitative, discussion based one. Additionally, we discussed the observations we made during the application of the patterns.

Data selection strategy The data was generated by the researchers taking part in this experiment. We had four researchers, who took part in this experiment including the one who designed the patterns. Two researchers were familiar with context patterns, and two were new to context pattern as well as the SOA topic.

Quality assurance, validity, and reliability As this experiment was only a feasibility study, no explicit means were taken to assure the quality. But one of the researchers taking part was an experienced professor.

A.1.4. Validation Case G: Experimental Simulation

Rationale The topic of this *experimental simulation* was to get a first impression whether the SOA (Layer / Stakeholder) Pattern were usable for someone who is not an expert in the field of SOA, and if the collected information is really helpful for conducting requirements engineering for a SOA-based system.

Purpose The experiment should provide insights whether a SOA setting can be described by a non domain expert using the SOA (Layer / Stakeholder) Pattern, and if the collected information was useful in a complete development cycle.

The case The case was the same case as used as running example in this work (See Chapter 5[5]).

[4] Page 75
[5] Page 75

The task for this experiment was to actually develop the system described using a rigid development process. The development was done by a student during his master thesis.

Research questions

- Are the SOA (Layer / Stakeholder) Pattern suitable for a non-domain expert to collect all important information in a structured form?
- Is the use of the SOA (Layer / Stakeholder) Pattern convenient for a non-domain expert?
- Is the information collected using the patterns suitable for improving downstream development activities?

Propositions

- The relationship between initial information, the information expected to be collected, and the information actually collected by a non-domain expert with the patterns was assessed.
- The relationship between the information collected using the patterns and the course of the actual development.

Method of data collection The data collected was of qualitative nature. The student documented his work in the master thesis. He was explicitly asked beforehand, to use the patterns and a defined development process, and to assess the usability for a SOA system, suggest improvements, find flaws and so forth. Some improvements were actually implemented by the student within his work. To track also the progress over time and not only the final product, the student was interviewed on a weekly basis.

Methods of data analysis The data analysis was a qualitative, discussion based one. Additionally, we discussed the observations we made during the application of the patterns.

Data selection strategy The data was generated by one student who implemented the SOA system, and one researcher observing his progress.

Quality assurance, validity, and reliability The final thesis was reviewed by three researchers including one not involved in the matter of context patterns at all. Additionally, three presentations took place, in which (intermediate) results were presented and discussed by the whole working group.

A.2. Cases Studies

Validation case B (Appendix A.2.1), case C (Appendix A.2.2), case D (Appendix A.2.3) and case H (Appendix A.2.4[6]) were conducted in the form of a case study.

A.2.1. Validation Case B: Case Study

Rational In this *case study*, we investigated whether the the Cloud System Analysis Pattern is suitable for a small enterprise to document its cloud infrastructure. This way, we tried to learn whether the Cloud System Analysis Pattern is useful and usable for such a company. The assumption behind this case study is that when such a small company can successfully

[6] Page 385

use a context pattern, the effort spend for using the context patterns can be spent by almost every company.

Purpose The case study served two purposes. First, to assess a context pattern in a real scenario without any influence of the authors of the context pattern. Second, to investigate if a company with very limited resources in money, time, and people is able to use the context pattern or not.

The case The case was a small enterprise offering a cloud infrastructure and cloud based services in the logistics domain. Here the Cloud System Analysis Pattern should be used for threat assessment and, finally, a ISO27000 certification. This case was part of a three years lasting project in which also tooling for the Cloud System Analysis Pattern and the ISO27000 documentation was developed. Beside our working group and the small enterprise, a second small enterprise specialized on certification, and another working group specialized on security assessments were involved in this project.

Research questions

- Can the Cloud System Analysis Pattern be applied without involving a context pattern expert?
- Is the Cloud System Analysis Pattern regarded as useful by a small enterprise?
- Is the Cloud System Analysis Pattern regarded as usable by a small enterprise?
- Is the effort spent for using the Cloud System Analysis Pattern regarded as reasonable?
- Can the Cloud System Analysis Pattern compete with current best practices in the field?
- Is the collected information suitable for a threat assessment and ISO27000 certification?

Propositions

- The relationship between the Cloud System Analysis Pattern and the perceived usability and usefulness of the results.
- The relationship between the Cloud System Analysis Pattern and the success of a ISO27000 certification

Method of data collection The data collected was the documentation produced for the certification, and feedback collected using unstructured interviews and discussions. The documentation was produced twice. The first time without the Cloud System Analysis pattern using best practice guides from the literature on the matter of ISO27000. The second time, using the Cloud System Analysis Pattern.

Methods of data analysis The data analysis was a qualitative one. We compared the different documentations. The results of this analysis were discussed with the project members. Additionally, we assessed the effort spent for producing each of the documentations, and the perceived usability and usefulness. This information was extracted from the interviews.

Data selection strategy The data was produced by the project members. There were two employees of the cloud provider, one person from our working group, one person from the second small enterprise, and one person from the second working group.

Quality assurance, validity, and reliability The generated documentation was part of the deliverable of the project as well as used afterward by the cloud provider. Hence, it was an integral part of the project activities to ensure the high quality of the documentation as well as the preciseness of the drawn conclusions. In consequence, the documents were reviewed and discussed several times by the project members. The context pattern authors were not involved in using the patterns, but only observed what was going on. Most analysis was done as post-mortem analysis to ensure that the authors could not influence the course of the documentation.

A.2.2. Validation Case C: Action Research

Rational In this *action research*, we replicated Case A (see Appendix A.1.1) in a real setting, because the results from Case A were promising but needed validation in a real case.

Purpose This action research should provide insights whether a cloud setting can be described using the Cloud System Analysis Pattern, the information described by the Cloud System Analysis Pattern can be used for further activities, and if the Law (Identification) Pattern work as intended. The practical purpose was to write a security and compliance concept for the system-to-be.

The case The case was a cloud platform developed by a research institute. The cloud platform offered several services and solutions for companies in the field of logistics. First cooperations with companies from the industry were already started, but for going productive, those companies were in need of a security and data protection concept for the used services, which did not included a detailed analysis of the covered requirements. For this case, the federal data protection act of Germany was of particular importance.

Research questions

- Is the Cloud System Analysis Pattern suitable to collect all important information in a structured form?
- Is the Cloud System Analysis Pattern convenient to use and the effort spent for using it reasonable?
- Is the collected information suitable for a law identification?
- Is the combined use of Law Pattern instances and Law Identification Pattern instances sufficient to identify a relevant law?
- Are the Law (Identification) Pattern convenient to use and the effort spent for using them reasonable?

Propositions

- The relationship between the use of the Cloud System Analysis Pattern and the sufficiency of the collected information for downstream activities.
- The relationship between the laws identified by the Law (Identification) Pattern and the laws documented after a legal revision.

Method of data collection The data collected was of qualitative nature. We conducted several meetings in which we used the patterns. Some workshops involved clients and/or

legal experts in which we presented and discussed our results. The results of the meetings were documented. For the compliance part, we also used a conventional discussion-based method to identify the relevant laws. The results of the pattern approach and the discussion-based legal revision were compared.

Methods of data analysis The data analysis was a qualitative, discussion based one. We compared the results with several experts of the domain and legal experts to find any insufficiencies. Additionally, the final judgment, regarding the produced concept, of the clients reviewers were analyzed.

Data selection strategy The data was generated by the researchers writing the concept.

Quality assurance, validity, and reliability The produced documentation was part of the deliverable to the clients. Hence, it was an integral part of the activities taken to ensure the high quality of the documentation as well as the preciseness of the drawn conclusions. In consequence, the documents were reviewed and discussed several times by the conducting researchers. The result was assessed by independent reviewers.

A.2.3. Validation Case D: Action Research

Rational In this *action research*, we had to understand the smart grid domain. For this domain, we had no context pattern. Hence, we used the Meta Pattern to derive a new one. The new pattern was the used to support the first steps of the Secure Development Live Cycle (SDLC) by Microsoft, which was used by the project partner.

Purpose The action research served two purposes. First, we investigated whether the Meta Pattern is helpful for deriving a new Context Pattern and if the meta model contained in the Meta Pattern is able to express all entities of the newly derived context pattern. Second, we assessed the sufficiency of the newly derived patterns for collecting the information as demanded by the initial steps of the SDLC.

The case One application domain of the NESSoS project, which we took part in, is the smart grid. Our task within this project was to deliver solutions to analyze the smart grid regarding security, privacy and compliance, for example with standards, in the early phase of the software development life cycle. In this particular case, the industrial partner was a world wide operating company focusing on the areas of electrification, automation and digitization. The topic of the case was to improve the early steps of the SDLC, as the industry partner was in need of more guidance for these steps.

Research questions

- Does the Meta Pattern support deriving new context patterns?
- Does the meta model contained in the Meta Pattern contain all elements to express a new context pattern?
- Is a context pattern derived for a domain using the Meta Pattern accepted by domain experts as valid?
- Is the newly derived pattern suitable to guide the initial steps of the SDLC?
- Is the effort spent for using the context patterns acceptable with regards to the effort spent using the former methods?

Propositions

- The relationship between the Meta Pattern and a newly derived context pattern which was not used as basis for the Meta Pattern.
- The relationship between a Context Pattern and the early steps of the SDLC.

Method of data collection The derived patterns (Smart Grid Pattern, and Smart Home Pattern) were part of the data. In workshops the initial patterns were discussed with experts from the client. The insights gained were documented. Afterward, the patterns were applied within the SDLC to one, already finished, case from the company, and also a new one for a prototype smart home. For applying the context patterns, also the time spent was tracked.

Methods of data analysis The context patterns themselves were assessed in a qualitative manner together with the partners experts. The results from the modified SDLC using the Smart Home Pattern were compared to the results from the unmodified SDLC. This assessment was done in a workshop style, and the differences and their severity for the development process were discussed. For the prototype smart home, only a qualitative discussion with domain experts were conducted.

Data selection strategy The data was generated by the project members. One employee of the project partner, and two persons from our working group were permanently involved. Others from the partners side as well as from the working groups side joined for some workshops.

Quality assurance, validity, and reliability The generated documentation was part of the deliverable of the EU project as well as used afterward by the partner. Hence, it was an integral part of the project activities to ensure the high quality of the documentation as well as the preciseness of the drawn conclusions. In consequence, the documents were reviewed and discussed several times by the project members, and also by reviewers who were not involved in this particular case but the NESSoS project itself.

A.2.4. Validation Case H: Case Study

Rational The topic of this *case study* was to get a evidence whether the SOA (Layer / Stakeholder) Pattern are suitable to improve the software development for a real system or not.

Purpose In this case an already existing system, which was built in an ad hoc manner, should be replaced with a new, systematically developed one. We were interested if the SOA (Layer / Stakeholder) Pattern help to structure the redevelopment and improve the outcome. For this case, we decided to not involve ourselves in the actual development, but be a pure observer to see how the patterns work in a real life setting, applied by the actual target audience of the patterns.

The case The case was provided by a student of ours who was also involved in a small size company providing tailored SOA based solutions for customer relationship and enterprise resource management. As it is often the case for start up companies, the development of the product was done in an unstructured, ad hoc manner. And they had chosen SOA for the benefits on the technology level, but were unaware of its implications. They also were missing an overview of stakeholders and their influence on the system. As the customer base was growing and also the usage of the old product, it became obvious to the small company that the established system was not maintainable on the long run as too many requirements were completely overlooked in the beginning. This led to the situation that

they planned to replace the old system by a complete rebuild. But they were really in need of a means which helped them to discover all the stakeholders and their relations as well as the technology elements and their relations, and a structured method to integrate the insights in a rigid development method. For this case, we provided the student a description of a development method he could use, and the patterns. He was then asked to use them and modify them as he wished to adapt them to his preferred agile development cycle.

Research questions

- Do the SOA (Layer / Stakeholder) Pattern help to find all important information and to structure the information?
- Is the collected information using the patterns suitable for improving downstream development activities which are conducted in an agile way?
- Are pattern users able to apply and use the patterns without any detailed guidance beyond the pure description?
- Is the resulting new system regarded as an improvement compared to the existing system?

Propositions

- The relationship between the information collected using the patterns and the information needed for the development activities was assessed.
- The relationship between the information collected using the patterns and the course of the actual agile development.

Method of data collection The data collected was of qualitative nature. The student documented his work in a master thesis. He was explicitly asked beforehand, to use the patterns and a defined development process, and to assess the usability for a SOA system, suggest improvements, find flaws and so forth. The student's topic was also to make modifications whenever needed to reflect his agile approach. To track also the progress over time and not only the final product, the student was interviewed on a weekly basis.

Methods of data analysis The data analysis was a qualitative, discussion based one. Additionally, we discussed the observations we made during the application of the patterns.

Data selection strategy The data was generated by one student who implemented the SOA system, and one researcher observing his progress.

Quality assurance, validity, and reliability The final thesis was reviewed by three researchers including one not involved in the matter of context patterns at all. Additionally, three presentations took place, in which (intermediate) results were presented and discussed by the whole working group.

A.3. Literature Reviews

Table A.1 shows the journals, conferences, and workshops which were manually searched for the different literature reviews. The baseline list was selected topic independently and contains journals, and conferences on the topic of software engineering in general, and requirements engineering in specific. The topic specific list was populated using a google search and terms

	Source	Abbreviation	Type	Level
Baseline List	Automated Software Engineering	ASE	Conference	A
	International Conference on Advanced Information Systems Engineering	CaiSE	Conference	A
	European Software Engineering Conference	ESEC	Conference	A
	International Conference on Software Engineering	ICSE	Conference	A
	International Requirements Engineering Conference	RE	Conference	A
	International Working Conference on Requirements Engineering: Foundation for Software Quality	REFSQ	Conference	B
	Symposium On Applied Computing (Track: Computer Security)	SAC:CS	Conference	B
	Symposium On Applied Computing (Track: Smart Grid and Smart Technologies)	SAC:SG	Conference	B
	Symposium On Applied Computing (Track: Software Engineering)	SAC:SE	Conference	B
	Symposium On Applied Computing (Track: Requirement Engineering)	SAC:RE	Conference	B
	Symposium On Applied Computing (Track: Service-Oriented Architectures and Programming)	SAC:SOA	Conference	B
	International Joint Conference on Software Technologies	ICSOFT	Conference	B
	ACM Transactions on Computer Systems	TOCS	Journal	A
	ACM Transactions on Software Engineering and Methodology	TOSEM	Journal	A
	IEEE Transactions on Software Engineering	TOSE	Journal	A
	Journal of Systems and Software	JSS	Journal	A
	Software and System Modeling	JSSM	Journal	B
	Requirements Engineering	RE-Journal	Journal	A
	Transactions on Pattern Languages of Programming	TPLOP	Journal	B
	International Conference on Availability, Reliability and Security	(CD-)ARES	Conference	B
	Europlop	EuroPLoP	Conference	B
	PloP	PloP	Conference	A
	International Symposium on Engineering Secure Software and Systems	ESSOS	Conference	none
	International Conference on Model Driven Engineering Languages and Systems	MODELS	Conference	B
Topic Specific List	Workshop on Modelling in Software Engineering	MiSE	Workshop	ICSE
	Software Engineering Challenges for the Smart Grid	SE4SG	Workshop	ICSE
	Formal Methods in Software Engineering	FormaliSE	Workshop	ICSE
	Principles of Engineering Service-Oriented Systems	PESOS	Workshop	ICSE
	General Theory of Software Engineering	GTSE	Workshop	ICSE
	Software Engineering for Secure Systems	SESS	Workshop	ICSE
	Software Quality	WOSQ	Workshop	ICSE
	International Workshop on Software Quality	SOQUA	Workshop	ESEC
	International Workshop on the Quality of Service-Oriented Software Systems	QUASOSS	Workshop	ESEC
	International Workshop on Governance, Risk and Compliance in Information Systems	GRISIS	Workshop	CaiSE
	International Workshop on Advances in Services DEsign based on the Notion of Capability	ASDENCA	Workshop	CaiSE
	Workshop on Information Systems Security Engineering	WISSE	Workshop	CaiSE
	Security and Privacy Requirements Engineering	ESPRE	Workshop	RE
	International Model-Driven Requirements Engineering Workshop	MoDRE	Workshop	RE
	International Workshop on the Interrelations between Requirements Engineering & Business Process Management	REBPM	Workshop	RE
	International Workshop on Requirements Patterns	REPA	Workshop	RE
	International Comparing *Requirements* Modeling Approaches Workshop	CMA@RE	Workshop	RE
	International Workshop on Requirements Engineering and Law	RELAW	Workshop	RE
	International Workshop on Managing Requirements Knowledge	MaRK	Workshop	RE
	International Workshop on Requirements Engineering for E-voting Systems	RE-Vote	Workshop	RE
	International Workshop on Requirements Prioritization and Communication	RePriCo	Workshop	REFSQ
	Smart Grid Security Workshop	SmartGridSec	Workshop	ESSOS
	Security Software Engineering	SecSE	Workshop	ARES
	Frontiers in Availability, Reliability and Security	FARES	Workshop	ARES
	International Conference on Human-Centered Software Engineering	HCSE	Conference	none
	Conference in Business Process Management	BPM	Conference	A
	International Workshop on Variability Modelling of Software-intensive Systems	VaMoS	Workshop	none
	International Workshop on Empirical Requirements Engineering	EmpiRE	Workshop	RE
	International Conference on Software Engineering and Knowledge Engineering	SEKE	Conference	B
	International Conference on Service Oriented Computing	ICSOC	Conference	A

Table A.1.: *List of Journals, Conferences and Workshops searched within Manual Search*

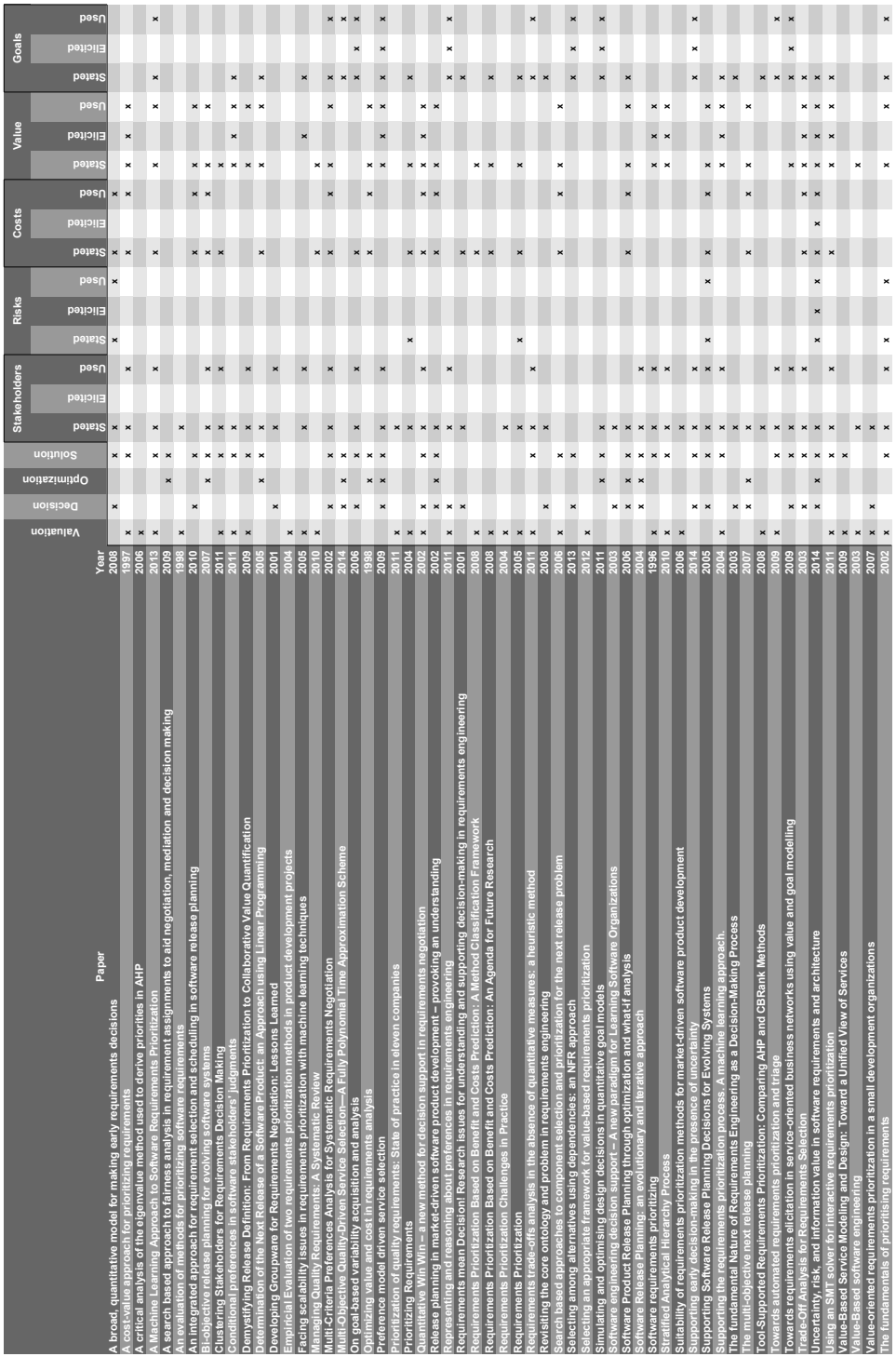
relevant for the topic at hand. Table A.1 contains the name of the source, the abbreviation of the sources, and its level. In case of a workshop the conference where the workshop takes place is named instead of the level.

Table A.2 shows the topics for which a google search was conducted (dark gray background). For each topic the search strings are given in **bold**. Underneath each search string the results, which were not already part of the list of sources, are given in *italics*.

Context Elicitation
context software engineering conference
International Conference on Human-Centered Software Engineering
context requirements engineering conference
workshop context software engineering
conference context requirements engineering
Process Elicitation
process management software engineering conference
Conference in Business Process Management
process management software engineering workshop
process management requirements engineering workshop
International Workshop on the Interrelations between Requirements Engineering & Business Process Management
process management requirements engineering conference
Security Requirements Elicitation
Security Requirements Elicitation software engineering conference
Security Requirements Elicitation software engineering workshop
Compliance Requirements Elicitation
Compliance Legal Law Requirements Elicitation conference
Compliance Legal Law Requirements Elicitation workshop
Interaction Detection
interaction detection software engineering conference
interaction detection software engineering workshop
Generation of Alternatives
requirements variability software engineering workshop
International Workshop on Variability Modelling of Software-intensive Systems
requirements variability software engineering conference
Valuation of Requirements
requirements valuation prioritization software engineering workshop
requirements valuation prioritization software engineering conference
Optimization of Requirements
requirements optimization software engineering conference
Symposium on Search-Based Software Engineering
requirements optimization software engineering workshop
International Workshop on Empirical Requirements Engineering
Smart Grid
smart grid requirements software engineering conference
smart grid requirements software engineering workshop
Service Oriented Architectures
Service Oriented Architectures requirements software engineering conference
International Conference on Software Engineering and Knowledge Engineering
International Conference on Service Oriented Computing
Service Oriented Architectures requirements software engineering workshop
Voting Systems
voting systems requirements software engineering conference
voting systems requirements software engineering workshop
Decision Making
decision making requirements software engineering conference
decision making requirements software engineering workshop
Pattern
pattern requirements software engineering conference
pattern requirements software engineering workshop

Table A.2.: *Search Terms and Results for Topic Specific Conferences And Workshops*

A.3.1. Decision Making and Requirements Selection in RE



Paper	Year	Valuation	Decision	Optimization	Solution	Stakeholders: Stated	Stakeholders: Elicited	Stakeholders: Used	Risks: Stated	Risks: Elicited	Risks: Used	Costs: Stated	Costs: Elicited	Costs: Used	Value: Stated	Value: Elicited	Value: Used	Goals: Stated	Goals: Elicited	Goals: Used
A broad, quantitative model for making early requirements decisions	2008		x		x	x			x		x	x		x	x		x			
A cost-value approach for prioritizing requirements	1997	x			x	x		x							x		x			
A critical analysis of the eigenvalue method used to derive priorities in AHP	2006	x																		
A Machine Learning Approach to Software Requirements Prioritization	2013	x			x	x		x				x			x		x	x		x
A search based approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making	2009			x	x															
An evaluation of methods for prioritizing software requirements	1998	x				x														
An integrated approach for requirement selection and scheduling in software release planning	2010		x		x							x		x	x		x			
Bi-objective release planning for evolving software systems	2007			x	x	x		x				x		x	x		x			
Clustering Stakeholders for Requirements Decision Making	2011	x			x	x		x				x			x		x	x		
Conditional preferences in software stakeholders' judgments	2011	x				x														
Demystifying Release Definition: From Requirements Prioritization to Collaborative Value Quantification	2009	x			x	x		x							x		x			
Determination of the Next Release of a Software Product: An Approach using Linear Programming	2005			x	x	x						x			x		x			
Developing Groupware for Requirements Negotiation: Lessons Learned	2001							x												
Empirical Evaluation of two requirements prioritization methods in product development projects	2004	x				x														
Facing scalability issues in requirements prioritization with machine learning techniques	2005	x																		
Managing Quality Requirements: A Systematic Review	2010	x				x						x			x			x		
Multi-Criteria Preferences Analysis for Systematic Requirements Negotiation	2002				x	x		x				x		x	x			x		x
Multi-Objective Quality-Driven Service Selection—A Fully Polynomial Time Approximation Scheme	2014				x	x														x
On goal-based variability acquisition and analysis	2006				x	x		x				x						x		x
Optimizing value and cost in requirements analysis	1998			x	x							x		x	x		x			
Preference model driven service selection	2009				x	x		x							x		x	x		x
Prioritization of quality requirements: State of practice in eleven companies	2011	x				x														
Prioritizing Requirements	2004	x				x			x			x			x			x		
Quantitative Win Win – a new method for decision support in requirements negotiation	2002	x		x	x	x		x				x		x	x		x			
Release planning in market-driven software product development – provoking an understanding	2002		x	x	x	x						x		x	x					
Representing and reasoning about preferences in requirements engineering	2011	x				x		x										x		x
Requirements mean Decision! Research issues for understanding and supporting decision-making in requirements engineering	2001		x			x						x						x		
Requirements Prioritization Based on Benefit and Costs Prediction: A Method Classification Framework	2006	x										x		x	x					
Requirements Prioritization Based on Benefit and Costs Prediction: An Agenda for Future Research	2008	x										x		x						
Requirements Prioritization Challenges in Practice	2004	x				x														
Requirements Prioritization	2005	x				x			x						x			x		
Requirements trade-offs analysis in the absence of quantitative measures: a heuristic method	2011	x				x		x			x							x		
Revisiting the core ontology and problem in requirements engineering	2008			x		x														
Search based approaches to component selection and prioritization for the next release problem	2006	x				x						x		x			x			
Selecting among alternatives using dependencies: an NFR approach	2013		x			x														
Selecting an appropriate framework for value-based requirements prioritization	2012	x																x		x
Simulating and optimising design decisions in quantitative goal models	2011				x	x												x		x
Software engineering decision support – A new paradigm for Learning Software Organizations	2003					x														
Software Product Release Planning through optimization and what-if analysis	2006				x	x						x		x	x					
Software Release Planning: an evolutionary and iterative approach	2004				x	x												x		x
Software requirements prioritizing	1996	x				x		x												
Stratified Analytical Hierarchy Process	2010	x				x		x							x		x			
Suitability of requirements prioritization methods for market-driven software product development	2006	x													x		x			
Supporting early decision-making in the presence of uncertainty	2014					x		x												
Supporting Software Release Planning Decisions for Evolving Systems	2005				x	x		x	x			x		x	x		x			
Supporting the requirements prioritization process. A machine learning approach.	2004	x				x		x							x		x			
The fundamental Nature of Requirements Engineering as a Decision-Making Process	2003					x						x			x					
The multi-objective next release planning	2007		x											x			x			
Tool-Supported Requirements Prioritization: Comparing AHP and CBRank Methods	2008	x				x												x		
Towards automated requirements prioritization and triage	2009	x						x												x
Towards requirements elicitation in service-oriented business networks using value and goal modelling	2009		x			x		x							x			x		x
Trade-Off Analysis for Requirements Selection	2003					x						x		x	x		x			
Uncertainty, risk, and information value in software requirements and architecture	2014		x	x		x			x	x		x		x	x		x			
Using an SMT solver for interactive requirements prioritization	2011	x				x		x				x			x		x			
Value-Based Service Modeling and Design: Toward a Unified View of Services	2009	x																		
Value-Based software engineering	2003	x				x									x					
Value-oriented requirements prioritization in a small development organizations	2007	x	x			x														
The fundamentals of prioritising requirements	2002	x													x		x			x

Table A.3.: *List of Relevant Papers Regarding Decision Making in RE (1/2)*

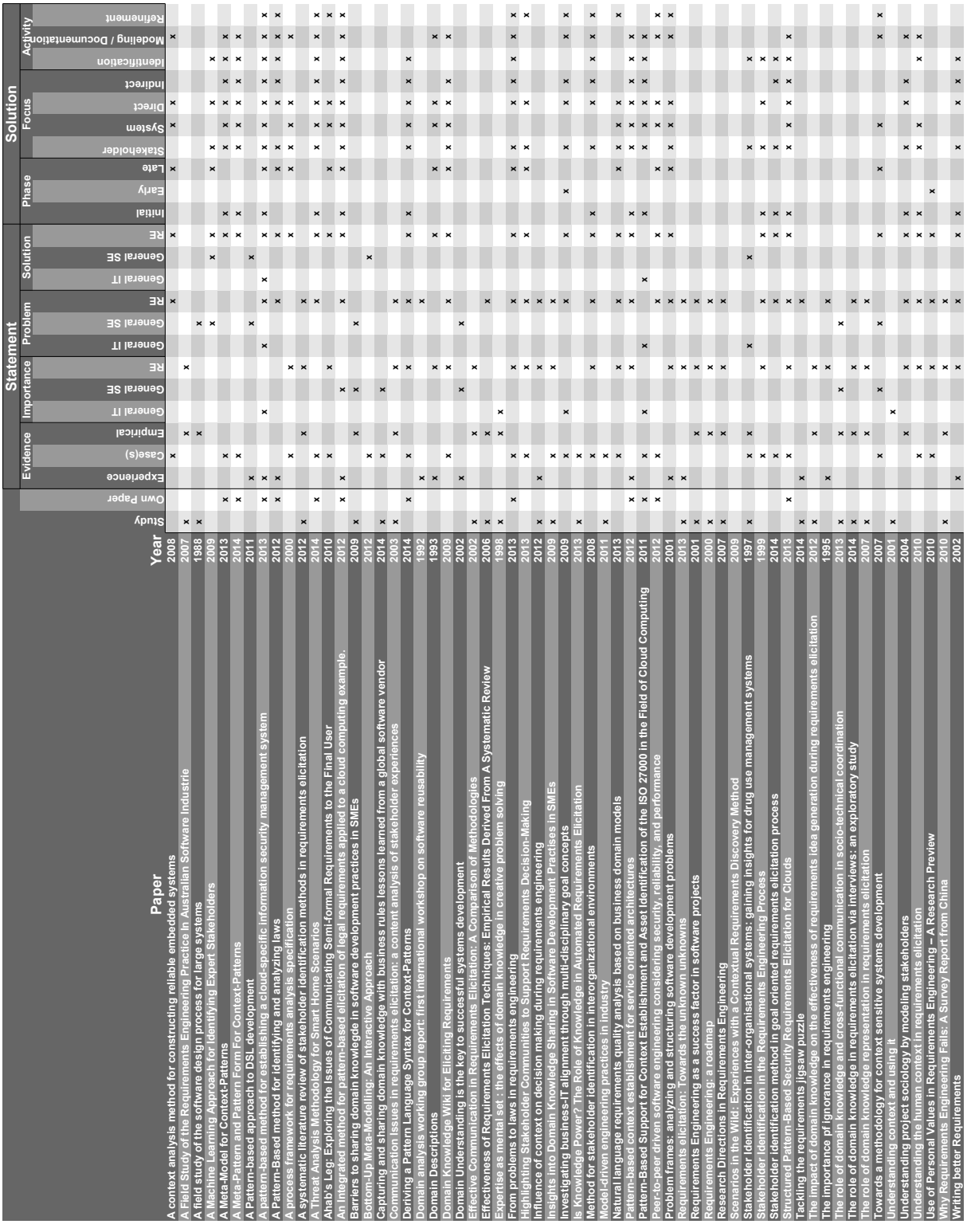
Table A.3 and Table A.4 show the papers selected as relevant for the literature review about decision making and requirements selection in RE. The first two columns contain the paper title and the publication year. The next three columns indicate whether the paper is on the topic of valuation, decision making, or optimization. The next column indicates whether the paper

describes a solution or not. The rest of the columns show the different kinds of information relevant for decision making. For each kind it is collected if the importance is stated, if it is explained how to elicit this information, and, in case the paper describes a solution, if the information is actually used.

		Processes			Relations			Context			Variability			Context Aggregated			Values Aggregated		
Paper	Year	Stated	Elicited	Used	Stated	Elicited	Used	Stated	Elicited	Used	Stated	Elicited	Used	Stated	Elicited	Used	Stated	Elicited	Used
A broad, quantitative model for making early requirements decisions	2008							x		x	x		x	x		x	x		x
A cost-value approach for prioritizing requirements	1998													x		x	x	x	x
A critical analysis of the eigenvalue method used to derive priorities in AHP	2006													x		x	x		x
A Machine Learning Approach to Software Requirements Prioritization	2013													x		x	x		x
A search based approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making	2009				x		x												
An evaluation of methods for prioritizing software requirements	1998													x					
An integrated approach for requirement selection and scheduling in software release planning	2010				x		x										x		x
Bi-objective release planning for evolving software systems	2007				x		x							x		x	x		x
Clustering Stakeholders for Requirements Decision Making	2011													x		x	x		x
Conditional preferences in software stakeholders' judgments	2011				x		x				x			x		x			
Demystifying Release Definition: From Requirements Prioritization to Collaborative Value Quantification	2009				x		x							x		x	x	x	x
Determination of the Next Release of a Software Product: an Approach using Linear Programming	2005	x			x									x			x		x
Developing Groupware for Requirements Negotiation: Lessons Learned	2001													x		x			
Empirical Evaluation of two requirements prioritization methods in product development projects	2004				x			x						x					
Facing scalability issues in requirements prioritization with machine learning techniques	2005													x		x		x	
Managing Quality Requirements: A Systematic Review	2010				x			x						x					
Multi-Criteria Preferences Analysis for Systematic Requirements Negotiation	2002										x			x		x	x		x
Multi-Objective Quality-Driven Service Selection—A Fully Polynomial Time Approximation Scheme	2014	x									x		x						
On goal-based variability acquisition and analysis	2006	x					x	x		x	x		x	x		x	x		x
Optimizing value and cost in requirements analysis	1998	x		x			x			x				x		x	x		x
Preference model driven service selection	2009	x		x	x		x							x		x	x	x	x
Prioritization of quality requirements: State of practice in eleven companies	2011				x			x						x					
Prioritizing Requirements	2004	x			x									x			x		
Quantitative Win-Win—a new method for decision support in requirements negotiation	2002				x			x			x		x			x	x	x	x
Release planning in market-driven software product development – provoking an understanding	2002				x		x							x		x	x		x
Representing and reasoning about preferences in requirements engineering	2011	x	x	x	x		x							x					
Requirements mean Decision! Research issues for understanding and supporting decision-making in requirements engineering	2001				x			x											
Requirements Prioritization Based on Benefit and Costs Prediction: A Method Classification Framework	2008				x									x					
Requirements Prioritization Based on Benefit and Costs Prediction: An Agenda for Future Research	2008				x									x					
Requirements Prioritization Challenges in Practice	2004																		
Requirements Prioritization	2005	x			x			x						x			x		
Requirements trade-offs analysis in the absence of quantitative measures: a heuristic method	2011				x		x							x		x			
Revisiting the core ontology and problem in requirements engineering	2008	x			x			x						x					
Search based approaches to component selection and prioritization for the next release problem	2006																x		x
Selecting among alternatives using dependencies: an NFR approach	2013				x		x				x								
Selecting an appropriate framework for value-based requirements prioritization	2012																		
Simulating and optimising design decisions in quantitative goal models	2011				x	x	x				x		x	x					
Software engineering decision support – A new paradigm for Learning Software Organizations	2003	x												x					
Software Product Release Planning through optimization and what-if analysis	2006	x			x		x							x			x		x
Software Release Planning: an evolutionary and iterative approach	2004				x		x							x		x			
Software requirements prioritizing	1996													x		x	x	x	x
Stratified Analytical Hierarchy Process	2010				x									x		x	x	x	x
Suitability of requirements prioritization methods for market-driven software product development	2006				x		x							x					
Supporting early decision-making in the presence of uncertainty	2014				x		x	x						x					
Supporting Software Release Planning Decisions for Evolving Systems	2005													x		x	x		x
Supporting the requirements prioritization process. A machine learning approach.	2004				x	x	x				x		x	x		x	x	x	x
The fundamental Nature of Requirements Engineering as a Decision-Making Process	2003	x			x					x				x					
The multi-objective next release planning	2007				x									x			x		x
Tool-Supported Requirements Prioritization: Comparing AHP and CBRank Methods	2008	x																	
Towards automated requirements prioritization and triage	2009	x		x	x		x							x		x			
Towards requirements elicitation in service-oriented business networks using value and goal modelling	2009	x			x		x							x		x	x	x	x
Trade-Off Analysis for Requirements Selection	2003				x			x			x		x	x		x	x	x	x
Uncertainty, risk, and information value in software requirements and architecture	2014	x									x		x	x		x	x	x	x
Using an SMT solver for interactive requirements prioritization	2011				x		x							x		x	x	x	x
Value-Based Service Modeling and Design: Toward a Unified View of Services	2009	x																	
Value-Based software engineering	2003	x												x			x		
Value-oriented requirements prioritization in a small development organizations	2007	x												x					
The fundamentals of prioritising requirements	2002				x		x							x		x			x

Table A.4.: *List of Relevant Papers Regarding Decision Making in RE (2/2)*

A.3.2. Context Elicitation



Paper	Year	Study	Own Paper	Statement: Evidence: Experience	Case(s)	Empirical	Statement: Importance: General IT	General SE	RE	Statement: Problem: General IT	General SE	RE	Solution: General IT	General SE	RE	Solution: Phase: Initial	Early	Late	Solution: Focus: Stakeholder	System	Direct	Indirect	Identification	Activity: Modeling / Documentation	Refinement
A context analysis method for constructing reliable embedded systems	2008				x				x									x		x				x	
A Field Study of the Requirements Engineering Practice in Australian Software Industrie	2007	x				x				x					x										
A field study of the software design process for large systems	1988	x				x		x																	
A Machine Learning Approach for Identifying Expert Stakeholders	2009				x													x							
A Meta-Model for Context-Patterns	2013		x		x									x		x				x	x	x			
A Meta-Pattern and Pattern Form For Context-Patterns	2014		x		x																				
A Pattern-based approach to DSL development	2011			x							x														
A pattern-based method for establishing a cloud-specific information security management system	2013		x	x						x		x		x	x			x		x	x	x		x	x
A Pattern-Based method for identifying and analyzing laws	2012		x	x								x		x				x		x	x	x		x	x
A process framework for requirements analysis specification	2000				x													x		x					
A systematic literature review of stakeholder identification methods in requirements elicitation	2012	x				x																			
A Threat Analysis Methodology for Smart Home Scenarios	2014		x							x		x		x	x			x		x	x	x			x
Ahab's Leg: Exploring the Issues of Communicating Semi-formal Requirements to the Final User	2010				x																				
An integrated method for pattern-based elicitation of legal requirements applied to a cloud computing example.	2012		x	x						x				x	x			x		x	x	x			x
Barriers to sharing domain knowlegde in software development practices in SMEs	2009	x				x					x														
Bottom-Up Meta-Modelling: An Interactive Approach	2012				x									x											
Capturing and sharing domain knowledge with business rules lessons learned from a global software vendor	2014	x			x																				
Communication Issues in requirements elicitation: a content analysis of stakeholder experiences	2003	x				x				x															
Deriving a Pattern Language Syntax for Context-Patterns	2014		x		x					x				x	x			x		x	x	x			
Domain analysis working group report: first international workshop on software reusability	1992			x						x															
Domain Descriptions	1993			x										x				x		x					x
Domain Knowledge Wiki for Eliciting Requirements	2009													x	x			x		x					x
Domain Understanding is the key to successful systems development	2002			x	x			x		x	x														
Effective Communication in Requirements Elicitation: A Comparison of Methodologies	2002	x				x																			
Effectiveness of Requirements Elicitation Techniques: Empirical Results Derived From A Systematic Review	2006	x				x			x																
Expertise as mental set : the effects of domain knowledge in creative problem solving	1998	x					x																		
From problems to laws in requirements engineering	2013			x	x					x		x		x	x			x		x	x	x			
Highlighting Stakeholder Communities to Support Requirements Decision-Making	2013				x					x		x		x	x			x		x					
Influence of context on decision making during requirements engineering	2012	x		x						x		x													
Insights into Domain Knowledge Sharing in Software Development Practises in SMEs	2009	x			x					x		x													
Investigating business-IT alignment through multi-disciplinary goal concepts	2009				x					x		x					x			x	x				
Is Knowledge Power? The Role of Knowledge in Automated Requirements Elicitation	2013	x			x									x	x										
Method for stakeholder identification in interorganizational environments	2008				x					x		x													
Model-driven engineering practices in industry	2011	x			x									x	x			x		x	x	x			
Natural language requirements quality analysis based on business domain models	2013				x					x		x													
Pattern-based context establishment for service oriented architectures	2012			x	x					x		x		x	x			x		x	x	x			
Pattern-Based Support for Context Establishment and Asset Identification of the ISO 27000 in the Field of Cloud Computing	2011			x	x								x												
Peer-to-peer driven software engineering considering security, reliability, and performance	2012			x						x		x		x	x			x		x	x	x			
Problem frames: analyzing and structuring software development problems	2001				x					x		x													
Requirements elicitation: Towards the unknown unknowns	2013	x			x					x		x													
Requirements Engineering as a success factor in software projects	2001	x							x																
Requirements Engineering: a roadmap	2000	x							x																
Research Directions in Requirements Engineering	2007	x							x																
Scenarios in the Wild: Experiences with a Contextual Requirements Discovery Method	2009																								
Stakeholder Identification in inter-organisational systems: gaining insights for drug use management systems	1997	x			x	x					x														
Stakeholder Identification in the Requirements Engineering Process	1999				x					x		x		x	x			x		x		x			
Stakeholder identification method in goal oriented requirements elicitation process	2014				x					x		x		x	x			x			x	x			
Structured Pattern-Based Security Requirements Elicitation for Clouds	2013			x	x					x		x		x	x			x		x	x	x			
Tackling the requirements jigsaw puzzle	2014	x																							
The impact of domain knowledge on the effectiveness of requirements idea generation during requirements elicitation	2012	x		x																					
The importance pf ignorance in requirements engineering	1995			x		x				x		x													
The role of domain knowledge and cross-functional communication in socio-technical coordination	2013	x				x					x														
The role of domain knowledge in requirements elicitation via interviews: an exploratory study	2014	x				x				x															
The role of domain knowledge representation in requirements elicitation	2007	x				x				x		x													
Towards a methodology for context sensitive systems development	2007										x														
Understanding context and using it	2001	x					x																		
Understanding project sociology by modeling stakeholders	2004					x				x				x	x			x		x					
Understanding the human context in requirements elicitation	2010				x					x				x	x					x					
Use of Personal Values in Requirements Engineering – A Research Preview	2010				x					x				x			x								
Why Requirements Engineering Fails: A Survey Report from China	2010	x				x					x														
Writing better Requirements	2002			x					x			x			x										

Table A.5.: *List of Relevant Papers Regarding Context Elicitation*

Table A.5 shows the papers selected as relevant for the literature review about context elicitation. The first two columns contain the paper title and the publication year. The next three

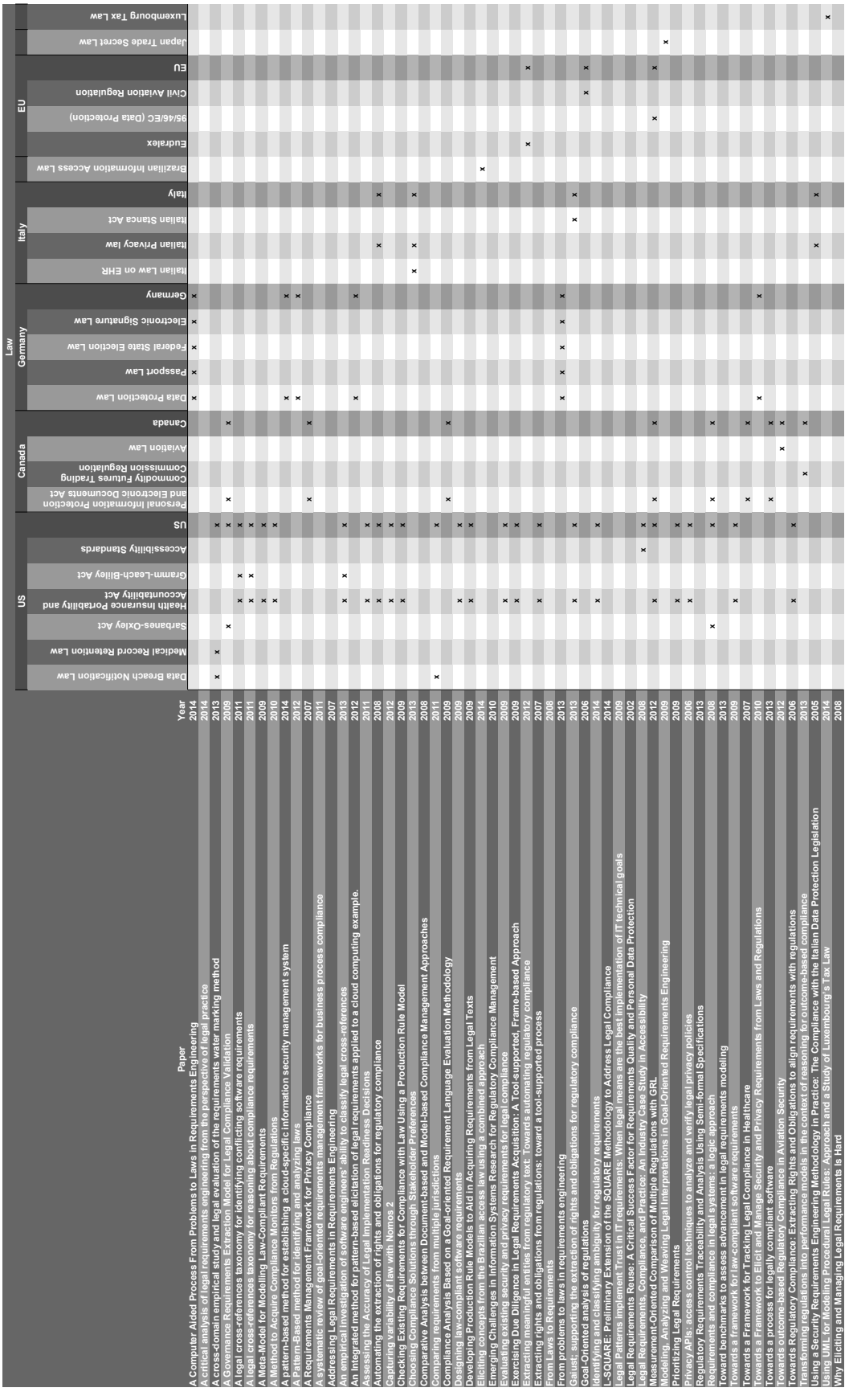
columns indicate on which evidence the statements given in the paper are based on. The three following columns indicate whether a statement about the importance of context elicitation is given in the paper and if this statement is valid for requirements engineering only, software engineering or IT in general. The next six columns indicate the same for problem statements and provided solutions. For those papers who provide a solution, the table also indicates which phase is covered, on which part of the system-to-be the solution focuses, and which activities are supported.

A.3.3. Compliance Requirements Elicitation

Table A.6and Table A.7 show the papers selected as relevant for the literature review about context elicitation. The first two columns contain the paper title and the publication year. The next two columns indicate whether the paper was written by us, and whether it describes a solution or not. The next three columns indicate the sub-phase covered by the paper. The ten following columns indicate the activity(ies) explained by the paper. The column after the activities indicates whether the solution explicitly integrates the collaboration with legal experts. This column is followed by three columns indicating the grade of formality of the solution describes by the paper. Finally, the next 27 columns indicate for which law(s) the solution was applied.

Paper	Year	Own	Solution	Sub-Phase: Initial	Sub-Phase: Early	Sub-Phase: Late	Activity: Modeling Law	Activity: Modeling Requirements	Activity: Elicitation	Activity: Reuse	Activity: Identification	Activity: Decision Making	Activity: Interaction Detection	Activity: Framework	Activity: Analysis Law	Activity: Analysis Requirements	Collaboration	Formality: Informal	Formality: Semi-Formal	Formality: Formal
A Computer Aided Process From Problems to Laws in Requirements Engineering	2014	x	x			x	x	x	x		x					x	x			
A critical analysis of legal requirements engineering from the perspective of legal practice	2014																		x	
A cross-domain empirical study and legal evaluation of the requirements water marking method	2013		x	x															x	
A Governance Requirements Extraction Model for Legal Compliance Validation	2009		x	x			x		x							x			x	
A legal cross-references taxonomy for identifying conflicting software requirements	2011		x			x							x						x	
A legal cross-references taxonomy for reasoning about compliance requirements	2011		x			x							x				x		x	
A Meta-Model for Modelling Law-Compliant Requirements	2009		x		x			x											x	
A Method to Acquire Compliance Monitors from Regulations	2010		x		x			x												x
A pattern-based method for establishing a cloud-specific information security management system	2014	x	x	x		x	x	x	x		x						x		x	
A Pattern-Based method for identifying and analyzing laws	2012	x	x	x		x	x	x	x		x						x		x	
A Requirements Management Framework for Privacy Compliance	2007		x		x			x						x					x	
A systematic review of goal-oriented requirements management frameworks for business process compliance	2011																			x
Addressing Legal Requirements in Requirements Engineering	2007																			x
An empirical investigation of software engineers' ability to classify legal cross-references	2013																			x
An integrated method for pattern-based elicitation of legal requirements applied to a cloud computing example.	2012	x	x	x		x	x	x	x		x						x		x	
Assessing the Accuracy of Legal Implementation Readiness Decisions	2011											x						x		
Automating the extraction of rights and obligations for regulatory compliance	2008		x			x			x							x				x
Capturing variability of law with Nomos 2	2012		x			x	x									x				x
Checking Existing Requirements for Compliance with Law Using a Production Rule Model	2009		x			x	x				x									x
Choosing Compliance Solutions through Stakeholder Preferences	2013		x		x						x									x
Comparative Analysis between Document-based and Model-based Compliance Management Approaches	2008											x								
Comparing requirements from multiple jurisdictions	2011		x			x										x				x
Compliance Analysis Based on a Goal-oriented Requirement Language Evaluation Methodology	2009		x		x											x			x	
Designing law-compliant software requirements	2009		x		x			x	x		x					x			x	
Developing Production Rule Models to Aid in Acquiring Requirements from Legal Texts	2009		x			x	x				x									x
Eliciting concepts from the Brazilian access law using a combined approach	2014		x		x		x												x	
Emerging Challenges in Information Systems Research for Regulatory Compliance Management	2010																			
Evaluating existing security and privacy requirements for legal compliance	2009		x			x										x			x	
Exercising Due Diligence in Legal Requirements Acquisition: A Tool-supported, Frame-based Approach	2009		x			x			x							x			x	
Extracting meaningful entities from regulatory text: Towards automating regulatory compliance	2012		x	x			x												x	
Extracting rights and obligations from regulations: toward a tool-supported process	2007		x	x					x							x			x	
From Laws to Requirements	2008											x								
From problems to laws in requirements engineering	2013	x	x			x	x	x	x		x					x	x		x	
Galust: supporting the extraction of rights and obligations for regulatory compliance	2013		x			x			x											
Goal-Oriented analysis of regulations	2006		x		x			x								x	x		x	
Identifying and classifying ambiguity for regulatory requirements	2014																			x
L-SQUARE: Preliminary Extension of the SQUARE Methodology to Address Legal Compliance	2014		x	x	x	x											x	x		
Legal Patterns implement Trust in IT requirements: When legal means are the best implementation of IT technical goals	2009		x		x			x						x					x	
Legal Requirements Reuse: A Critical Success Factor for Requirements Quality and Personal Data Protection	2002		x			x				x								x		
Legal Requirements, Compliance, and Practice: An Industry Case Study in Accessibility	2008																	x		
Measurement-Oriented Comparison of Multiple Regulations with GRL	2012		x		x		x						x		x				x	
Modeling, Analyzing and Weaving Legal Interpretations in Goal-Oriented Requirements Engineering	2009		x		x			x											x	
Prioritizing Legal Requirements	2009		x		x							x					x			x
Privacy APIs: access control techniques to analyze and verify legal privacy policies	2006		x	x	x	x	x								x					x
Regulatory Requirements Traceability and Analysis Using Semi-formal Specifications	2013		x	x				x	x										x	
Requirements and compliance in legal systems: a logic approach	2008		x			x										x				
Toward benchmarks to assess advancement in legal requirements modeling	2013																			x
Towards a framework for law-compliant software requirements	2009		x		x			x						x					x	
Towards a Framework for Tracking Legal Compliance in Healthcare	2007		x		x			x						x					x	
Towards a Framework to Elicit and Manage Security and Privacy Requirements from Laws and Regulations	2010		x		x									x					x	
Towards a process for legally compliant software	2013		x			x	x	x								x				x
Towards outcome-based Regulatory Compliance in Aviation Security	2012		x	x			x								x				x	
Towards Regulatory Compliance: Extracting Rights and Obligations to align requirements with regulations	2006		x	x					x				x			x			x	
Transforming regulations into performance models in the context of reasoning for outcome-based compliance	2013		x		x		x								x				x	
Using a Security Requirements Engineering Methodology in Practice: The Compliance with the Italian Data Protection Legislation	2005		x		x			x											x	
Using UML for Modeling Procedural Legal Rules: Approach and a Study of Luxembourg's Tax Law	2014		x	x			x				x									x
Why Eliciting and Managing Legal Requirements is Hard	2008																			

Table A.6.: *List of Relevant Papers Regarding Legal Compliance*



Paper	Year	US						US	Canada		Law Germany				Italy				EU					
		Data Breach Notification Law	Medical Record Retention Law	Sarbanes-Oxley Act	Health Insurance Portability and Accountability Act	Gramm-Leach-Bliley Act	Accessibility Standards	US	Personal Information Protection and Electronic Documents Act	Commodity Futures Trading Commission Regulation	Aviation Law	Canada	Data Protection Law	Passport Law	Federal State Election Law	Electronic Signature Law	Germany	Italian Law on EHR	Italian Privacy law	Italian Stanca Act	Italy	Brazilian Information Access Law	Eudralex	95/46/EC (Data Protection)	Civil Aviation Regulation	EU	Japan Trade Secret Law	Luxembourg Tax Law
A Computer Aided Process From Problems to Laws in Requirements Engineering	2014																										
A critical analysis of legal requirements engineering from the perspective of legal practice	2014																										
A cross-domain empirical study and legal evaluation of the requirements water marking method	2013	x	x					x																			
A Governance Requirements Extraction Model for Legal Compliance Validation	2009			x				x	x			x															
A legal cross-references taxonomy for identifying conflicting software requirements	2011				x	x		x	x																		
A legal cross-references taxonomy for reasoning about compliance requirements	2011				x	x		x	x																		
A Meta-Model for Modelling Law-Compliant Requirements	2009																										
A Method to Acquire Compliance Monitors from Regulations	2010																										
A pattern-based method for establishing a cloud-specific information security management system	2014																										
A Pattern-Based method for identifying and analyzing laws	2012																										
A Requirements Management Framework for Privacy Compliance	2007																										
A systematic review of goal-oriented requirements management frameworks for business process compliance	2011																										
Addressing Legal Requirements in Requirements Engineering	2007																										
An empirical investigation of software engineers' ability to classify legal cross-references	2013																										
An Integrated method for pattern-based elicitation of legal requirements applied to a cloud computing example.	2012																										
Assessing the Accuracy of Legal Implementation Readiness Decisions	2011																										
Automating the extraction of rights and obligations for regulatory compliance	2008																										
Capturing variability of law with Nomos 2	2012																										
Checking Existing Requirements for Compliance with Law Using a Production Rule Model	2009																										
Choosing Compliance Solutions through Stakeholder Preferences	2013																										
Comparative Analysis between Document-based and Model-based Compliance Management Approaches	2008																										
Comparing requirements from multiple jurisdictions	2011																										
Compliance Analysis Based on a Goal-oriented Requirement Language Evaluation Methodology	2009																										
Designing law-compliant software requirements	2009																										
Developing Production Rule Models to Aid in Acquiring Requirements from Legal Texts	2009																										
Eliciting concepts from the Brazilian access law using a combined approach	2014																										
Emerging Challenges in Information Systems Research for Regulatory Compliance Management	2010																										
Evaluating existing security and privacy requirements for legal compliance	2009																										
Exercising Due Diligence in Legal Requirements Acquisition: A Tool-supported, Frame-based Approach	2009																										
Extracting meaningful entities from regulatory text: Towards automating regulatory compliance	2012																										
Extracting rights and obligations from regulations: toward a tool-supported process	2007																										
From Laws to Requirements	2008																										
From problems to laws in requirements engineering	2012																										
Galust: supporting the extraction of rights and obligations for regulatory compliance	2013																										
Goal-Oriented analysis of regulations	2006																										
Identifying and classifying ambiguity for regulatory requirements	2014																										
L-SQUARE: Preliminary Extension of the SQUARE Methodology to Address Legal Compliance	2014																										
Legal Patterns Implement Trust in IT requirements: When legal means are the best implementation of IT technical goals	2009																										
Legal Requirements Reuse: A Critical Success Factor for Requirements Quality and Personal Data Protection	2002																										
Legal Requirements, Compliance, and Practice: An Industry Case Study in Accessibility	2008						x	x																			
Measurement-Oriented Comparison of Multiple Regulations with GRL	2012						x	x				x													x		
Modeling, Analyzing and Weaving Legal Interpretations in Goal-Oriented Requirements Engineering	2009																							x			
Prioritizing Legal Requirements	2009						x	x																		x	
Privacy APIs: access control techniques to analyze and verify legal privacy policies	2006						x	x																			
Regulatory Requirements Traceability and Analysis Using Semi-formal Specifications	2013																										
Requirements and compliance in legal systems: a logic approach	2008			x				x	x			x															
Toward benchmarks to assess advancement in legal requirements modeling	2013																										
Towards a framework for law-compliant software requirements	2009				x			x																			
Towards a Framework for Tracking Legal Compliance in Healthcare	2007											x															
Towards a Framework to Elicit and Manage Security and Privacy Requirements from Laws and Regulations	2010								x				x				x										
Towards a process for legally compliant software	2013								x			x															
Towards outcome-based Regulatory Compliance in Aviation Security	2012										x	x															
Towards Regulatory Compliance: Extracting Rights and Obligations to align requirements with regulations	2006				x			x																			
Transforming regulations into performance models in the context of reasoning for outcome-based compliance	2013									x		x															
Using a Security Requirements Engineering Methodology in Practice: The Compliance with the Italian Data Protection Legislation	2005																		x								
Using UML for Modeling Procedural Legal Rules: Approach and a Study of Luxembourg's Tax Law	2014																										x
Why Eliciting and Managing Legal Requirements Is Hard	2008																										

Table A.7.: *List of Relevant Papers Regarding Legal Compliance (2/2)*

Appendix B

Context Elicitation

This appendix contains the rest of the context patterns, which were not already shown in Chapter 8[1] (Appendix B.1), as well as detailed information about the relations between different context patterns (Appendix B.2[2]).

[1] Page 131
[2] Page 446

B.1. Catalog of Context Patterns

B.1.1. Meta Pattern

The meta model (see Section 7.3[3]) describes the common entities and relations of context patterns. The meta process (see Section 7.4[4]) describes the commonalities in deriving and describing a context pattern. The meta process and the meta model form the solution of our Meta Pattern, which is described in the following. (Note that the Meta Pattern repeats content of the sections about the meta model and meta process for reasons of completeness) The context Meta Pattern was validated by applying it to the smart grid domain, which was not part of the set of context patterns, and therefore domains, used for deriving the meta model and meta process.

	Pattern Name	Meta Pattern **Classification** Meta Pattern
	Related Patterns	-
Summary	**Intent**	A context pattern solves the problem of elicitation and structured description of context information for a particular domain. The Meta Pattern can be used to describe a context pattern.
	Essence	When doing software engineering, it might happen that the software engineers have to deal with and understand a particular domain and the context given by the domain. In many cases, the software engineers are not familiar with this particular domain. And there might be reasons that they also have to document the insights they gained about the domain for downstream activities. But as the domain is unknown, the information which should be collected, and the sufficient degree of abstraction and detail for this information is unknown. *Deriving a new context pattern, if none is already available, can help to solve the problems in this situation. The Meta Pattern contains the information about sources that are useful for deriving a context pattern, elements a context pattern should contain, and a process how to derive and validate a context pattern.*
	Known Uses	• Cloud System Analysis Pattern [37, 45, 44, 39, 274] • Law Pattern [45, 43, 44, 133, 134] • Law identification Pattern [45, 43, 44, 133, 134] • P2P Pattern [37, 38] • Smart Grid Pattern [49] • Smart Home Pattern [51] • SOA Layer Pattern [41] • SOA Stakeholder Pattern [41]
Motivation	**Example**	Once, we were involved in a cloud project in which the general compliance of a cloud system should be assessed in the very early stages of software development. For this task, a clean and precise understanding and description of the context was necessary. The reasons for having a context description were manifold.

[3]Page 120
[4]Page 126

Motivation

Example

In the field of standards, such as the common criteria or ISO 27001, one needs to have a context description to enable certification assessors to judge your system. When investigating the compliance of a system to laws, the context of a system is a key information as the applicability of a law is strongly context sensitive. In the same way, to judge if a requirement is fulfilled by the system, regardless which regulation or stakeholder is its source, one needs to know the environment (context) of the system.
The problems we were facing in this situation were diverse. First, we had to know which the key elements are for describing the context of a cloud. Which information at which detail is necessary to have a context description to be informative, comprehensible, and useful for the certification assessors, legal experts and requirements engineers? How to collect this information in a structured way? How to organize the information? How to avoid information overloading? Unfortunately, the domain space of cloud computing is very large and incoherent. And knowledge about the clouds exists on very different levels of abstraction starting from high level concepts down to technical details. And we were no cloud experts. But only talking to some domain experts and involved project members turned out to be insufficient as it seemed that there was no common ground in the understanding of the cloud domain and every expert had his/her own view on (parts of) the cloud domain and system which should be assessed.

Context

Context information elicitation is a general problem when dealing with IT systems. Often the IT experts who have to analyze, asses, implement, run, or maintain a system are not experts in the domain(s) where the system is used. Moreover, even when they know the domain, big parts of the required knowledge is tacit knowledge within an organization. Hence, they do not posses the detailed knowledge about the environment of the IT system. And for a complex IT system the environment is also very complicated and contains diverse legacy systems, people interacting with the system, people influencing the system, and so forth. All these elements bring their own constraints, goals, knowledge, and so forth. But when coming to, for example, compliance, standards, laws or internal regulations, or requirements engineering, there is a demand for clean and precise understanding or description of the context of the IT system. And there should be a structured and reusable way to solve this problem.

Problem

1. Which information has to be collected in a domain which is unknown?
2. How to represent and persist the important elements of a domain found while analyzing the domain for (re)use?
3. How to collect knowledge about the important elements in a structured way especially when it is tacit knowledge?
4. How to externalize and store the collected information that it is useful afterward?
5. How to find the right level of abstraction of the context description?
6. How to avoid information overloading by providing the minimum but complete information the target information consumer needs?

Motivation

Forces

There are several factors which have an influence on the context elicitation that make it complicated to find a solution:

No coherent and widely accepted definition of a domain available For many domains, there is not one central definition. In most cases, there are several commonly accepted descriptions of domains such as scientific publications, standards, regulations, surveys, and so forth. Those sources differ in some points as they often have a different view on a domain or use a different wording. The commonalities, and therefore main elements, in these documents have to be discovered and the wording needs to be aligned.

Knowledge about important elements is scattered, diverse, and tacit within an organization For collecting the information one needs to know where to collect it. In many cases, this is the first challenge as there are several places and people where this information resides. Again, the wordings are often not aligned, hence, a mapping is needed. And the different persons have a different view on the important elements. Hence, the view of all those people needs to be harmonized. And their tacit knowledge needs to be externalized and connected. All these challenges make a communication-heavy process necessary. For this process, means which support and structure the communication are needed.

Applicability and usability If a context pattern will be used afterward and the gain achieved by using it outweighs the effort spent, strongly depends on the usability of the pattern. And the usability is not automatically given when usefulness is already proven. The result might be regarded as useful but the elicitation and description process might be to tedious to justify the usage of the context pattern.

Needs of the target information consumer The sufficiency of a context description for its purpose strongly depends on the needs of the ones who have to understand and use it. These needs dictate the level of abstraction, information detail, and representation.

Solution

Structure

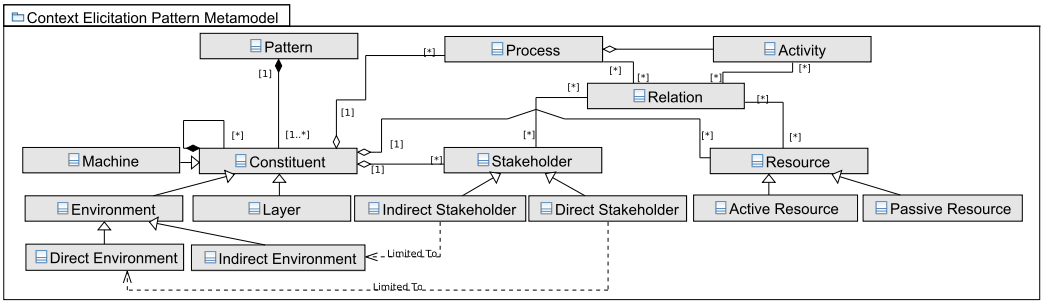


Fig. B.1: Important entities and relations for the Meta Pattern

Solution

Structure

Table B.2 Information Collection Template Pattern for Stakeholders

Name What is the name or identifier of the stakeholder?

Description Which important properties does the stakeholder have? What characterizes the stakeholder? What is its place in the environment?

Motivation Which objectives does the stakeholder follow? Why does the stakeholder influence the organization(s) / provider(s)?

Top Level Goals Which top level goals does the stakeholder have?
☐ **Adaptability** ☐ **Compliance** ☐ **Economy** ☐ **Efficiency**
☐ **Evolvability** ☐ **Learnability** ☐ **Maintainability** ☐ **Modularity**
☐ **Performance** ☐ **Portability** ☐ **Privacy** ☐ **Reliability** ☐ **Resilience**
☐ **Re-Usability** ☐ **Robustness** ☐ **Safety** ☐ **Scalability** ☐ **Security**
☐ **Testability** ☐ **Understandability** ☐ **Usability**

Kind
☐ **Specific** Is the stakeholder a real entity? Is the stakeholder not used to represent a group?
☐ **Representative** Is the stakeholder a real existing entity? Is this stakeholder used as proxy for a group of homogeneous stakeholders?
☐ **Group** Is the stakeholder not a real existing entity? Is this stakeholder used to describe for a group of homogeneous stakeholders?
☐ **Role** Can this stakeholder be shared through groups of heterogeneous stakeholders? Are there well-defined rights and permissions for this stakeholder?

Entities

Constituent Each pattern contains at least one constituent. In general, an constituent contains elements of either a technical or organizational view. A constituent can contain other constituents, which do not need to be of the same view.

Machine A constituent concerned with the machine, i.e. the thing to be developed

Environment The environment is a constituent which in turn contains elements that have some kind of relation to the machine.

Direct Environment There are elements which directly interact with the machine, captured in the direct environment.

Indirect Environment There are elements which have an influence on the system via elements of the direct environment, captured by the indirect environment.

Layer A layer is a constituent which encapsulates a set of elements within the environment or a machine. The elements are of the same level regarding some defined order.

Solution

Entities

Process A process describes some kind of workflow or sequence of activities.

Activities activities are a part of a process and in which certain tasks and actions have to be carried out.

Stakeholder A stakeholder describes a person, a group of persons, or organizational units, which have some kind of influence on the machine.

Direct Stakeholder Direct stakeholder interacts directly with the machine.

Indirect Stakeholder Indirect stakeholder who only interacts with direct stakeholders but has some interest in or influence on the machine.

Resource A resource describes some physical or non-physical (e.g., information) element which is needed to run the machine or which is processed by the machine and which is not a stakeholder.

Active Resource A resource can be an active resource with some behavior.

Passive Resource A resource can be a passive resource without any behavior.

Collaborations

Relation Several relations are possible between processes, activities, stakeholders and resources.

Method

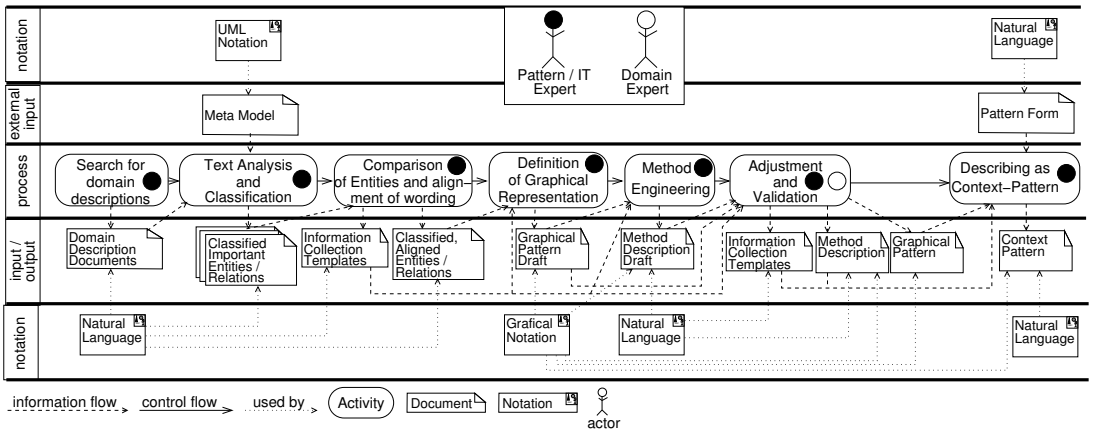


Fig. B.2: Process Pattern for Using the Meta Model to derive a context pattern.

The method is conducted by *pattern / IT experts* and *domain experts* (see Fig. B.2). The method starts with *searching for documents* describing the unknown domain. Such documents can be domain standards, technical documents, or scientific or white papers. There should be as many *domain description documents* as possible, which are central for a domain and of good quality, but at least three for the subsequent steps.

Solution

Method

Once these documents are collected, the text and contained figures have to be analyzed for important entities. At this point, the *meta model* can guide the pattern expert. Reoccurring terms can be collected and mapped to the meta model. This way, a first impression of the basic semantic of a term is captured and entities are formed. Moreover, the meta model gives guidance for which terms one should search.

After all entities and relations are collected from the documents in isolation, the *classified important entities / relations* have to be *compared and the wording needs to be aligned*. Entities with the same semantic are grouped and one specific term to name them is selected. Grouping elements is comparable to the course of abstracting different occurrences to the concept they share. Hence, one can use different abstraction strategies as described by Baumgartner and Kohls (2013 [33]). A combination of segregation and aggregation strategy might be sufficient here, which means that one removes all detailed information which is not relevant for a comparison in a first step. In a second step one removes all attributes which are not shared for the elements at hand and checked whether a common, not trivial structure (such as everything has a name) remains. This way, a coherent set of *classified and aligned entities and collaborations* is derived from the different documents describing the domain under investigation.

While classifying the entities, one should also track which information is given about the entities in the source documents. Information which is always given or requested for an entity in all source documents is described using so called *information collection templates*. These templates serve later on, when the context pattern is used for context elicitation, as kind of questionnaire to collect information about an entity. The common information collection template for stakeholders is shown in Table B.2.

Next, the classified and aligned entities and collaborations between them are *expressed and visualized using a graphical representation*. As already discussed in Section 6.3, combining graphical representations which give an quick overview augment the comprehensibility of information given textually. Hence, the graphical representation should only capture the most important information about a domain to keep the graphical representation understandable. All other information should be stored in the according information collection templates. In the end, one should have a *graphical pattern draft* regardless which notation is used. The next activity is *method engineering*. For this activity, we use the agenda principle as described by Heisel (1998 [178]). An agenda describes a sequence of activities, with necessary inputs, the desired outputs, actors, the notations used, and validation conditions to check coherence. We accompanied the purely textual representation using agenda diagrams. Figure B.2 is an instance of such a diagram. One can use whatever he/she wants to describe processes. For us the agenda principle turned out to be useful. The output of the activity method engineering is a *method description draft*. All drafts should be *adjusted and validated* in discussions with domain experts. The final versions of the *information collection templates*, the *method description*, and the *graphical pattern* are the result of this activity. Last, all final results are compiled to a context pattern description using a *pattern form*.

B.1.2. Law Pattern

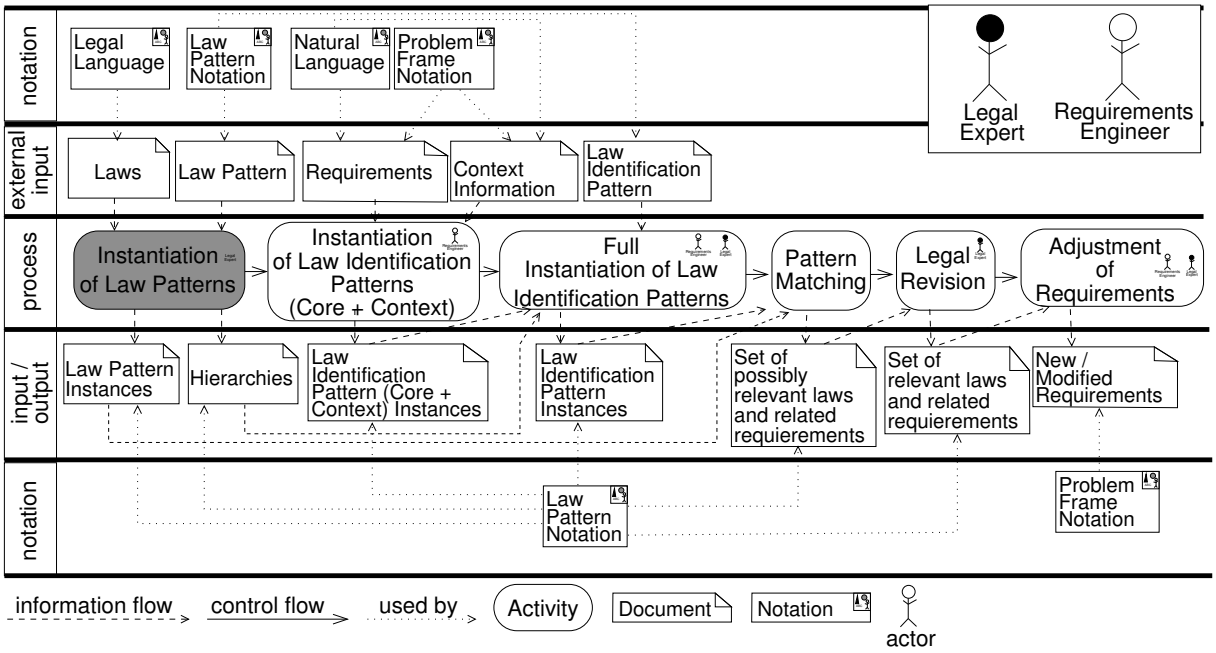


Figure B.3.: *General Process for Identifying Laws based on Requirements : Law Pattern Instantiation*

The problem of law identification is recurring for each system-to-be while the solution used by legal experts remains the same. Hence, we propose a pattern-based method for identifying and analyzing laws. The patterns allow the identification of relevant laws for a system-to-be based on its requirements. The full process for the identification is shown in Fig. B.3. The Law Pattern is of special concern for the first step of this process (highlighted in gray). The full discussion on law identification can be found in Chapter 14[5] Chapter 15[6], and Chapter 16[7].

We analyzed how judges and lawyers are supposed to analyze a law, based upon legal literature research and discussions with legal experts. These insights lead to a basic structure of laws and the contained sections, which we used to create a law pattern.

Pattern Name	Law Pattern	**Classification**	Organizational
Related Patterns	**Law Identification**		
Summary **Intent**	The worlds of software engineering and laws are two distinct worlds with very different wordings, methods, and culture. Nevertheless, software systems have to be compliant to laws. This pattern describes how to extract and document the essence of laws in a way that they can be used in software engineering in general and in requirements engineering in particular. This pattern is applicable for statute laws. Hence, it is not directly applicable for case law.		
Summary **Essence**	The wording used in the legal domain and the software engineering domain are very different. So are the methods to analyze a system of interest. Nevertheless, legal experts and software engineers have to cooperate as IT systems have to be compliant to the laws of the jurisdiction they operate in. The first problem to face in such a cooperation is to understand the important parts of the laws under consideration and the wording the laws embody.		

[5]Page 239
[6]Page 267
[7]Page 281

Summary

Essence

Without such an understanding, the legal and software engineering world cannot be aligned. *Read the structural part of our Law Pattern to understand the essence of laws and the context they form. Afterward, instantiate this pattern for all laws which might be of interest. The solution helps you to elicit all essential information within laws in a structured way and not to forget relevant parts. The Law Pattern and the accompanying method also describe how to document the resulting information. While describing a law you also set up different hierarchies of words which are important within the law. The resulting Law Pattern instances and hierarchies are a brief description of the essence of law, enabling a quick understanding what is of importance within a law. Additionally, the information is documented in a way it is usable in software engineering activities, such as requirements engineering.*

Known Uses

- Methoden und Techniken der Rechtsanwendung (English translation of the title: "Methods and techniques for the application of the law") [35]
- Einführung in das juristische Denken[8] [129]
- Methodenlehre der Rechtswissenschaft[9] [230]
- Juristische Methodik mit Technik der Fallbearbeitung[10] [344]
- Juristische Methodenlehre [11] [402]
- Several case described by legal experts in discussions

Motivation

Example

Once, we were involved in a cloud project in which the general compliance of a cloud system should be assessed in the very early stages of software development. For this task, a clean and precise understanding of the relevant laws was necessary. While the system development itself had already started, there was basically no information about the compliance of this system.
The problems we were facing in this situation were diverse. We had not even a clue which laws might be of relevance for our case. And when looking into laws, it was hard to understand which parts are important for judging its relevance. When talking to legal experts, we got a deeper understanding for a specific law discussed, but even then it was an open question how to document the gained information to be useful afterward. And in many cases the discussions were very specifically tailored to one particular question. Hence, the gained information was not helpful for judging any other question.
The situation was complicated by the intentional ambiguity of laws and the used wording. This made it hard for us to relate a concept given in a law to our actual system. Sometimes the used wording was even misleading as the use within a law was different of the use in daily live. And the discussions with legal experts were often fruitless as the legal experts have a kind of different thinking and different ways how to analyze a case, switching between different views on statements given within a law. One further aspect which complicates the understanding of laws are the manifold relations between different laws, between sections within a law, and even between words. Often one particular section seems to allow a certain action in general, but refers to another section which disallows this action.

[8] English translation of the title: "Introduction to legal thinking"
[9] English translation of the title: "Methodology of the jurisprudence"
[10] English translation of the title: "Legal Methodology with the technique for case processing"
[11] English translation of the title: "Legal Methodology"

Motivation

Context

In general, every legislator demands from everyone who lives in or is active within the jurisdiction of the legislators to comply with the laws the legislator enacts. Hence, software engineers have to assure that the system to be developed is compliant to all relevant laws of the jurisdiction the system-to-be will touch. Therefore, they need to know the legal requirements for the system to be developed. Based on this knowledge, the engineers can decide whether and how to address compliance.
Beside the general legal context and the resulting financial and criminal penalties, there are also economic reasons for companies to take up and deal with the legal compliance topic. Recent studies show that both, compliance violations (Cavusoglu et al., 2004 [98]) and insufficient preparations regarding upcoming regulations (Khansa et al., 2012 [213]), have a negative impact on the market share and value. Moreover, transparency regarding specific compliance topics, for example data protection, gain more and more importance as incentives to convince customers of and investors for an IT product and give a competitive edge against competitors (Zwick, 2006 [403]; Vehlow and Golkowsky, 2010 [381]; Unabhängiges Landeszentrum für Datenschutz Schleswig-Holstein, 2014 [375]).
This pattern is applicable statute law. Hence, it is not directly applicable for case law.

Problem

The general problem of describing the essence of a law is centered around different questions:

1. Which parts of a law are of relevance for judging its relevance for a case?
2. How to find the correct level of abstraction for the essence of a law in such a way that it is applicable to many different cases and not only a specific one?
3. How to document the essence of a law so that it is valuable for downstream activities?

Forces

There are several factors, which have an influence on the description of a law, making it complicated to find a solution:

The way of thinking and working of legal experts and software experts The communication between legal experts and software experts, which is unavoidable for describing laws, can get very complicated and tedious. The working style in both domains is very different, so is the way of communicating and discussing.

Ambiguity in laws Every law has the aim to be as general as possible to cover as many future cases as possible. Laws are not changed frequently. Hence, the legislator tries to formulate them in such a way that they are still applicable after years. The result is a very abstract formulation of legal texts, and an intended ambiguity in formulations as well as in wording. When describing a law, this ambiguity has to be considered. But not by removing the ambiguity, which is not possible without introducing errors as the ambiguity is there by intention and there exists no unambiguous and correct formulation or word.

Wording in laws Every law defines its own wording. Often, the use of commonly known words within a law is completely different from the daily life use. To find the important words for a law and capture their use within the law is not an easy task.

Motivation **Forces**

Laws, parts of laws, or words are often related Laws, parts of laws, or even words cannot be analyzed in isolation. Laws can be related in such a way that one law extends another law, or that laws are even competing in their application to a case leading to conflicting legal requirements. Hence, these relations are of great importance. This also true for the sections and statements within a law. Often, a section is incomplete and completed by other sections, for example exceptions to one section are given in another section, or the legal consequences of one section are refined within another section. And even the important words of a law are related forming a hierarchy. All these relations are of great importance when documenting a law, but can also be confusing when they are not treated systematically.

Solution **Structure**

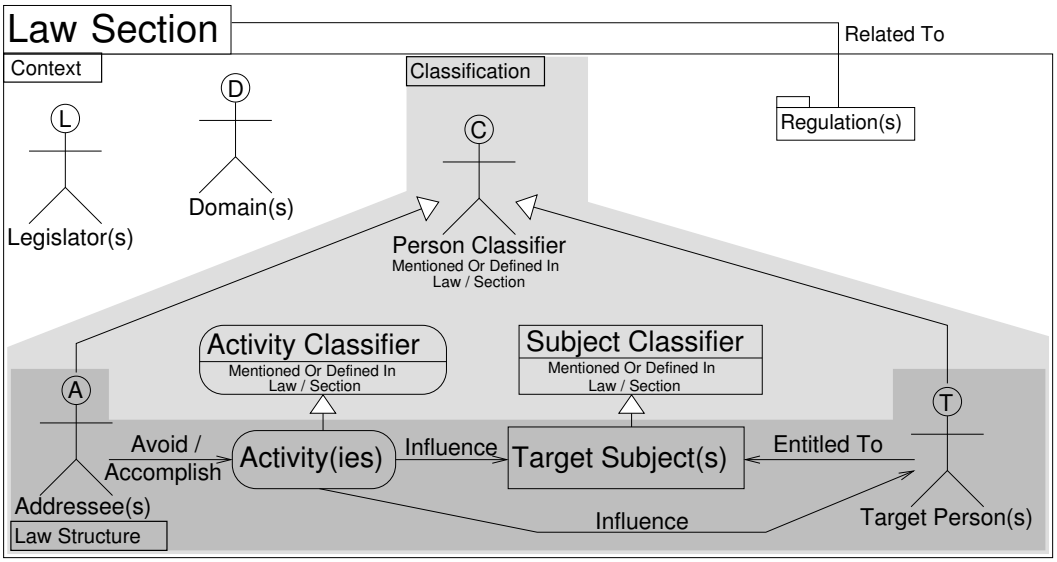


Fig. B.4: Law Pattern

The German law is a *statute law* in the tradition of the Roman jurisdiction. Statute laws are specified by the legislator and written down in legal documents. Hence, every judgment of a court is based exclusively on the analysis of the legal documents relevant for the judged case (Schwacke, 2003 [344], p. 41).
First of all, a law is a textual document. This law document is structured into *sections*. Each section defines a legal aspect of the law and contains several statements. These statements are *dictates of justice*, so-called *legal rules* (Larenz, 1983 [230], p. 240). There are different types of dictates of justice. Complete and self-contained dictates of justice are one type. This type is the fundamental building block of every law (Larenz, 1983 [230], p. 241).
A Complete and self-contained dictate of justice is divided into the *facts of the case*, the setting which is regulated, and the *legal consequence*, the resulting implications of the setting. Furthermore, a dictate of justice has also an *addressee(s)*.
The facts of the case need to be further refined to be useful for a pattern later on. The legal method called *subsumption* contains a further refinement of the facts of the case. This refinement results in the *law structure elements activities*, *target subjects*, and *target persons* (Beaucamp and Treder, 2011 [35], pp. 23-31).

Table B.4 Structure of a Full Dictate of Justice

Addressee(s)	has (have) to comply to the law.	
Facts of the case	**Activity(ies)**	describe(s) actions that an addressee has to follow or avoid to be compliant.
	Target subject(s)*	describes impersonal subjects that are objectives of the activity(ies). Subjects can be material, such as a product, or immaterial, such as information.
	Target person(s)*	are directly influenced by the activity(ies) of an addressee, or have a relation to the target subject(s).
Legal consequence	defines the consequence for an addressee, e.g. the punishment when violating the section.	

A * next to an element of the structure means that the element is optional.

Besides the complete, self-contained dictates there are: (Larenz, 1983 [230], pp. 247-251)

- *definition dictates* that describe and refine terms and other basic elements.
- *restricting dictates*, which add exceptions to a complete dictate
- *directing dictates*, which reference one or more other dictates. The referenced dictates contain (parts of) the law structure relevant for the dictate at hand.
- *fiction dictates*, which equate different facts of the case. The equated facts have to be treated as similar for judging a case even though they are different.

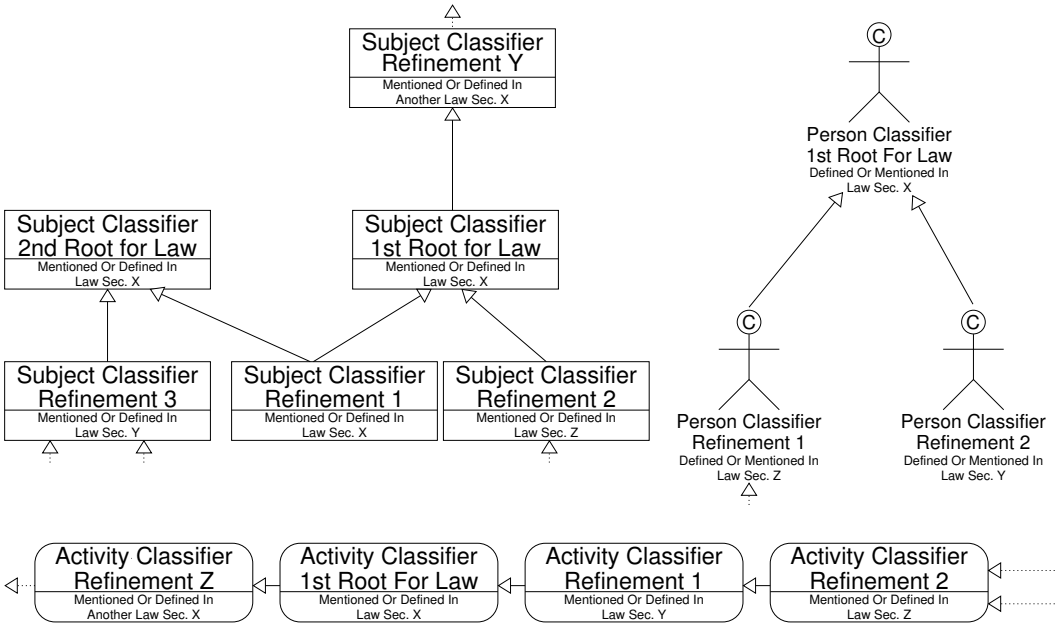


Fig. B.5: Classifier Hierarchies

Solution

Structure

All of these dictates cannot be analyzed in isolation, as they have relations to other dictates (or even laws). The types of relation between these dictates are *refinement*, *addition*, and *constraint*. This implies that all of the resulting dictates and laws, and the relations between them, have to be considered when analyzing laws. A *regulation* is the set of rules applicable to a specific case (Larenz, 1983 [230], p. 254).
Thus, relations between laws, sections and dictates of justice are of fundamental importance. They are arranged in a hierarchy, which is not always free of conflicts (Larenz, 1983 [230], p. 255). A special part of these relations is the terminology used within a jurisdiction. This terminology is organized as a tree where the terms of the more general dictates of justices are refined by subsequent dictates of justice. Figure B.5 shows a generic example for such term hierarchies. One can distinguish three basic kinds of terms. Subjects, which are depicted as rectangles, persons, which are depicted as stick-figures, and activities which are depicted as ellipses. Terms of these three kinds are used to formulate the important elements of a dictate of justice.
We organize the already mentioned hierarchies of person classifier, activity classifier, and subject classifier using a tree structure. Figure B.5 shows the generic structure of such hierarchies. A law structure element can refine another law structure element of the same type. For example, the *person classifier "refinement 1"* refines the *person classifier "1st root for law"*. While this example shows a refinement within one law, it is also possible that a law structure element refines a law structure element of another law. Such a case is the *activity classifier "1st root for law"* which refines the *activity classifier "refinement z"*. "refinement z" is part of another law. A law structure element is a root law structure element for a law if it does not refine any other element of the law. It is possible to have several root law structure elements of the same type for a law. For example, *"1st root for law"* and *"2nd root for law"* are both *subject classifiers* and a root law structure element for the same law. It is also possible that a law structure element refines more than one other law structure element. For example, the *subject classifier "refinement 1"* refines *"1st root for law"* and *"2nd root for law"*.

Entities

Direct Environment **Indirect Environment**

Context Contains all elements which are not of direct relevance if a dictate of justice applies to a case, but which are of relevance to judge if a law as a whole has to be considered.

Law Structure Contains the elements of the dictate of justice which have to be matched by the facts of the case. If the facts of the case match the elements, the dictate of justice might be relevant.

Solution Entities — Constituent | Indirect Stakeholder | Direct Stakeholder | Stakeholder | Activity | Resource

Classification Contains the elements which are needed to build the different term hierarchies for persons, activities and subjects which are part of every law.

Legislator The legislator describes the government of, for example, a country. A legislator enacts and enforces the law at hand.

Domain The domain represents the target audience for which the law applies. In the most general case, its the general public, but it might also only apply for a certain, for example, industry.

Addressee Every dictate of justice defines one or more stakeholder(s) who must comply to the dictate.

Target Person A target person is involved in the case but has a passive role. He/she is important to judge if a dictate of justice is applicable, but in most cases has not to comply with it.

Person Classifier The terms used for addressees and target persons form a hierarchy. Every law starts with generic person classifiers defining the general case for which it is applicable. These terms are refined afterward as the dictates of justice in later sections are more specific.

Activity(ies) describe(s) actions that an addressee has to follow or avoid to be compliant.

Activity Classifier The terms used for activities form a hierarchy. Every law starts with generic activity classifiers defining the general case for which the law is applicable. These terms are refined afterward as the dictates of justice in later sections are more specific.

Target Subject describes impersonal subjects. Subjects can be material, such as a product, or immaterial, such as information.

Subject Classifier The terms used for target subjects form a hierarchy. Every law starts with generic subject classifiers defining the general case for which the law is applicable. These terms are refined afterward as the dictates of justice in later sections are more specific.

Regulations Some dictates of justice are not self-contained as they refer to other dictates of justice, for example for some facts of the case. Such other regulations can be dictates of justice in the same law, but also dictates of justice in other laws.

Solution

Collaborations

Relation

Avoid / Accomplish Relates the addressee with the activity(ies) he / she must do (accomplish) or must not do (avoid) to comply to the dictate of justice at hand.

Influence Relates activity(ies) with the target subject(s) or person(s) which are influenced by the activity(ies).

Entitled To Relates target person(s) with target subject(s). Such an relation exists whenever target person(s) have certain rights regarding the target subject(s).

Is A Relates elements of the same type (activities, subjects, persons). The terms used within laws form a hierarchy. Hence, each element of a law structure, for example an addressee, is a refinement of another already defined element. The "Is A"-relation is used to express this information.

Method

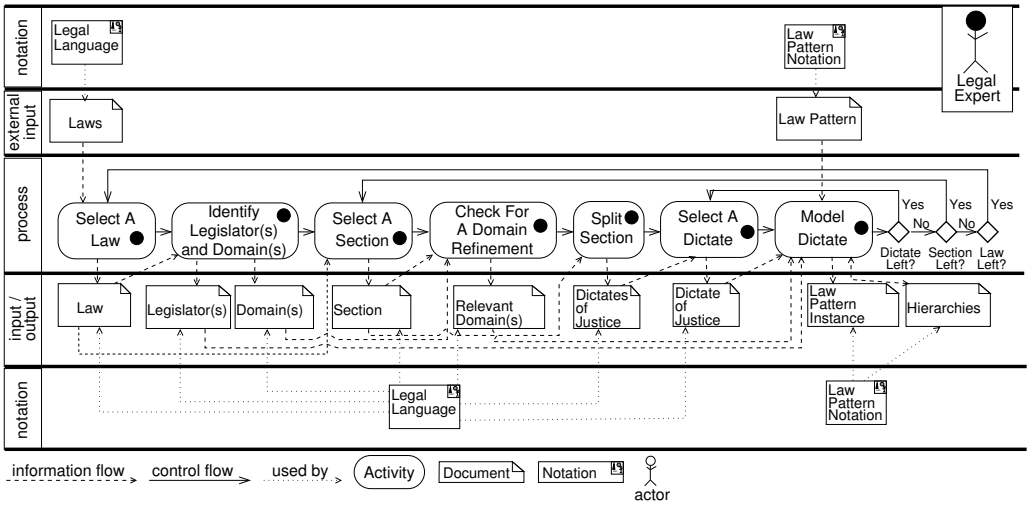


Fig. B.6: Law Pattern Instantiation Process

The process of instantiating Law Patterns, as depicted in Fig. B.6, starts with a set of *laws* which have to be modeled. From this set, we *select a law* which is not modeled yet. For this *law*, we *identify* the according *legislator(s)* and the target *domain(s)*. The information about the legislator(s) is given in the preamble of the law text. Sometimes also the *domain(s)* are given in the preamble. In case the domain(s) are not defined in the preamble, they are defined in the first section containing a complete dictate at the latest. In some case no specific domain is mentioned. Then, the domain "general public" applies. The identified legislator(s) and domain(s) apply for all dictates of justice contained within a law. But it is possible that for some sections of the law the domain, to which the law is applied, is refined. Hence, we assume for all sections the same domain(s) unless we find an explicitly mentioned refinement, which then applies only for this particular section. Next, we *select a section* of the law at hand which was not modeled up to this point. Note that sections can contain subsections. For selecting the next section we do not distinguish between top level sections and subsections.

We iterate over the sections in the order they occur in the document. For the selected *section*, we *check for a domain refinement*. In case we find such a refinement, we consider the refined domain(s) as *relevant domains* for the section at hand. In case we do not find a refinement, the domain(s) given by the laws are considered as relevant domains.
As a section can contain several *dictates of justice*, we *split the section* at hand into these dictates of justice. Then we *select* one *dictate of justice* which is not already modeled and proceed further. For the dictate of justice at hand we have to determine its type. The next activities to be taken vary depending on the type of the dictate of justice.

Instantiation of a Definition Dictate of Justice

For a definition dictate of justice we *identify* the contained *law structure element(s)* as next step. Hence, we look for terms or formulations within the text of the dictate of justice which indicate *addresses, activities, target subjects, or target persons*. Then, we *add the new elements to the according element hierarchy*. This includes that we also add the refinement relations which might be implied by the dictate of justice at hand.

Instantiation of a Fiction Dictate of Justice

For modeling, a fiction dictate of justice can be treated like a definition dictate of justice. Within the legal revision the definition and fiction dictates are treated differently. But this has no influence on the modeling.

Instantiation of a Complete Dictate of Justice

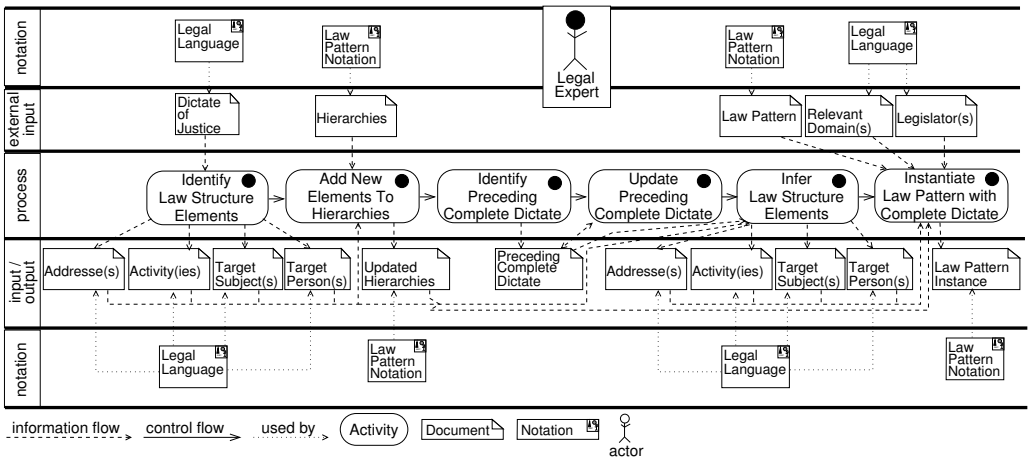


Fig. B.7: Model Dictate (Complete Dictate of Justice)

For a complete dictate of justice we *identify* the contained *law structure element(s)* as next step (see Fig. B.7). Hence, we look for terms or formulations within the text of the dictate of justice which indicate *addresses, activities, target subjects, or target persons*. For all found law structure elements we check if they are already part of the according element hierarchies or if they are newly introduced elements. In the latter case, we *add the new elements to the according element hierarchy*. This includes that we also add the inheritance relations which might be implied by the dictate of justice at hand.

Solution **Method**

Then, we *identify the preceding complete dictate of justice*. Most complete dictates of justice are refinements of other complete dictates of justice. Thus, these complete dictates of justice just add additional information or circumstances which detail aspects of the more general case. The most general complete dictate of justice is defined in the first section of a law. The *preceding complete dictate of justice* is explicitly mentioned in the dictate of justice at hand or it is the last complete dictate of justice defined in the preceding sections. The preceding dictate and the dictate of justice at hand are related regulations. Hence, we have to *update* the related regulations for the *preceding dictate of justice*.
In most cases, a complete dictate of justice which refines another complete dictate of justice only refines some elements. Thus, it might not contain all needed law structure elements explicitly. Hence, we have to *infer the missing elements* from the preceding complete dictate of justice. With inferring the missing elements we get the *complete law structure element(s)*.
In the last step, we use these elements, together with the relevant regulations, domain(s), and legislator(s), to *instantiate the complete dictate of justice*. The classification part can be directly taken from the according element hierarchies.

Instantiation of a Referring Dictate of Justice

Referring dictates of justice not only contain references within one law but also define references to other laws. Thus, we have to *identify the referenced law and section* and check whether the law, which is referenced by the referring dictate of justice, is already modeled or not. In case it is not already modeled, we have to *add the law* and the particular section which is referenced. The newly added law might be a candidate to be fully modeled in the future.
Then we *identify the connected complete dictate(s) of justice*. A referring dictate always adds further information to a complete dictate of justice (*the connected dictate of justice*). This might be information about competing regulations, which have to be discussed, complementing regulations, which have to be obeyed at the same time as the connected dictate of justice, substitutional dictates of justice which can be obeyed instead of the connected dictate of justice, or just a further dictate of justice which contains additional law structure elements or legal consequences. The referred law and section, the dictate of justice at hand and the connected dictate of justice are related regulations. Hence, we have to *update* the related regulations for the *connected dictate of justice*.

Instantiation of a Restricting Dictate of Justice

For adding a restricting dictate of justice, we *identify the restricted complete dictate of justice*. A restricting dictate of justice does not define a law structure, but adds restrictions to the core structure of a complete dictate of justice. Sometimes it also adds additional legal consequences. The *restricted complete dictate justice* is explicitly mentioned in the dictate of justice at hand or it is the last complete dictate of justice defined in the preceding sections. The restriction of the law structure or the additional legal consequence are not of relevance for finding relevant laws, but are of relevance for the legal revision and adjustment of requirements. Hence, we have to *update* the related regulations for the *restricted dictate of justice* as the restricted dictate and the dictate of justice at hand are related regulations.

B.1.3. Law Identification Pattern

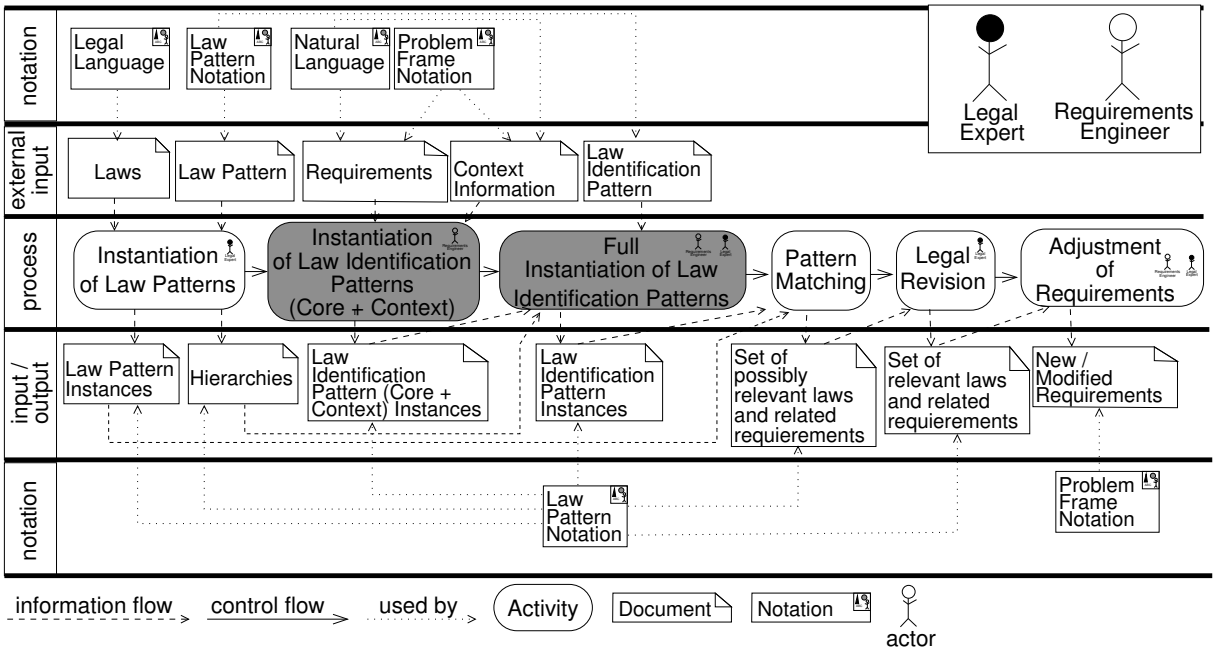


Figure B.8.: *General Process for Identifying Laws based on Requirements : Law Identification Pattern Instantiation*

The problem of law identification is recurring for each system-to-be while the solution used by legal experts remains the same. Hence, we propose a pattern-based method for identifying and analyzing laws. The patterns allow the identification of relevant laws for a system-to-be based on its requirements. The full process for the identification is shown in Fig. B.8. The Law Identification Pattern is of special concern for the second and third step of this process (highlighted in gray). The full discussion on law identification can be found in Chapter 14[12] Chapter 15[13], and Chapter 16[14].

We analyzed how judges and lawyers are supposed to analyze a case under legal consideration, based upon legal literature research and discussions with legal experts. These insights lead to a basic structure, which should be used for describing a system in such a way it is useful to determine whether a law is applicable or not.

Summary				
	Pattern Name	Law Identification Pattern	**Classification**	Organizational
	Related Patterns	***Law Identification***, *Smart Grid*, *Smart Home*, *Cloud System Analysis*, *SOA Stakeholder*		
	Intent	The worlds of software engineering and laws are two distinct worlds with very different wordings, methods, and culture. Nevertheless, software systems have to be compliant to laws. This pattern describes how to structure and document information about a system-to-be in a way that they can be used in for a legal analysis in software engineering in general and in requirements engineering in particular.		

[12]Page 239
[13]Page 267
[14]Page 281

Summary

Essence

The wording used in the legal domain and the software engineering domain are very different. So are the methods to analyze a system of interest. Nevertheless, legal experts and software engineers have to cooperate as IT systems have to be compliant to the laws of the jurisdiction they operate in. One problem to face in such a cooperation is to understand how the system-to-be should be described so that the information is a sufficient case description and therefore facilitates the legal analysis. One specific problem tackled by the pattern is how to bridge the differences in wording of the legal and the software engineering domain. Without such an understanding, the legal and software engineering world cannot be aligned. *Read the structural part of our Law Identification Pattern to understand which information about a system-to-be is important for a legal analysis. Afterward, instantiate this pattern for different parts of the description of the system-to-be which might be of interest. The solution helps you to gather all essential information about a system-to-be in a structured way and not to forget relevant parts. The Law Identification Pattern and the accompanying method also describe how to document the resulting information. While documenting a particular requirement you also relate words of the domain of the system-to-be to words of the legal domain. The resulting Law Identification Pattern instances are a brief description of the essence of a system-to-be, enabling a quick understanding what is of importance for a legal analysis. Additionally, the essence is be based on information which is used in software engineering activities, such as requirements engineering.*

Known Uses

- Methoden und Techniken der Rechtsanwendung[15] [35]
- Einführung in das juristische Denken[16] [129]
- Methodenlehre der Rechtswissenschaft[17] [230]
- Juristische Methodik mit Technik der Fallbearbeitung[18] [344]
- Juristische Methodenlehre[19] [402]
- Several cases described by legal experts in discussions[20]

Motivation

Example

Once, we were involved in a cloud project in which the general compliance of a cloud system should be assessed in the very early stages of software development. For this task, a clean and precise understanding of the relevant laws was necessary. The system development itself had already started, but there was basically no information about the compliance of this system.
The problems we were facing in this situation were diverse. We had not even a clue which information about the system was relevant for assessing the relevance of different laws. When talking to legal experts, we got a deeper understanding for a specific part of the system discussed, but this discussions were often tedious in the beginning as we did not provide the information the legal experts needed in a way they could understand.

[15] English translation of the title: "Methods and techniques for the application of the law"
[16] English translation of the title: "Introduction to legal thinking"
[17] English translation of the title: "Methodology of the jurisprudence"
[18] English translation of the title: "Legal Methodology with the technique for case processing"
[19] English translation of the title: "Legal Methodology"
[20] Note that the Law Pattern and the Law Identification Pattern are based on the same literature and discussions, as the literature and discussions are about the structure of laws as well as about how to structure cases for a law analysis.

Motivation

Example

And in many cases the discussions were very specific and tailored to one particular question. Hence, the gained information was not helpful for judging any other question. One particular problem within the discussions was the aligning of words of the system domain and words of the legal domain. It often happened that a discussion was leading nowhere until we discovered that we had a differing understanding of important words.
The situation was complicated by the intentional ambiguity of laws and the used wording. This made it hard for us to relate a concept given in a law to our actual system. Sometimes the used wording was even misleading as the use within a law was different of the use in daily live. And the discussions with legal experts were often fruitless as the legal experts have a different kind of thinking and different ways how to analyze a case, switching between different views on a specific part of the system. One more aspect which complicates the assessing of the relevance of a law are the manifold relations between different requirements, processes and so forth. In the beginning, we were unaware of the fact that, for example, some requirements, when assessed in isolation, might not be regulated by a law, but a certain combination of these requirements are regulated.

Context

In general, every legislator demands from everyone who lives in or is active within its to comply with the laws the legislator enacts. Hence, software engineers have to assure that the system to be developed is compliant to all relevant laws of the jurisdiction the system-to-be will touch. Therefore, they need to know the legal requirements for the system to be developed. Based on this knowledge, the engineers can decide whether and how to address compliance.
Beside the general legal context and the resulting financial and criminal penalties, there are also economic reasons for companies to take up and deal with the legal compliance topic. Recent studies show that both, compliance violations (Cavusoglu et al., 2004 [98]) and insufficient preparations regarding upcoming regulations (Khansa et al., 2012 [213]), have a negative impact on the market share and value. Moreover, transparency regarding specific compliance topics, for example data protection, gain more and more importance as incentives to convince customers of and investors for an IT product, and give a competitive edge against competitors (Zwick, 2006 [403]; Vehlow and Golkowsky, 2010 [381]; Unabhängiges Landeszentrum für Datenschutz Schleswig-Holstein, 2014 [375]).

Problem

The general problem of describing the essence of a law is centered around different questions:

1. Which parts of a system description are of relevance for judging if a law is relevant?
2. How to find correct level of abstraction for the essence of a system description in way that it is assessable for many different laws and not only a specific one?
3. How to document the essence of a system-to-be so that it is valuable for the assessment of the relevance of a law?
4. How to bridge the gaps between specific words of the system domain and words of the legal domain?

Motivation

Forces

There are several factors, which have an influence on the description of a system-to-be in a way that the description is suitable for assessing its compliance to relevant laws, making it complicated to find a solution:

The way of thinking and working of legal experts and software experts The communication between legal experts and software experts, which is unavoidable for assessing the legal compliance of a system, can get very complicated and tedious. The working style in both domains is very different, so is the way of communicating and discussing.

Wording used in the domain of the system is too specific and technical Software engineers are used to define and use a very precise wording for a system. This can be confusing for the legal experts, as from the abstract level, which is used in the legal domain, many system specific words refer to same legal term. Moreover, for the legal experts it is hard to conceive the general, abstract concept of technical words, but to identify the the general, abstract concept is one key to a successful compliance assessment.

Too much information about the system-to-be In a normal case, there is a vast body of information about a complex system. To assess each and every piece of information is not possible, and in general not necessary, as carving out some basic characteristics of the system-to-be is sufficient for finding relevant laws.

Many dependencies between parts of the information When assessing a system for its legal compliance, legal experts have to have a holistic view on the system. Therefore, it is not sufficient to look at pieces of information about the system in isolation. The relations and dependencies between system artifacts and pieces of information have to be made explicit and thoroughly considered.

Solution

Structure

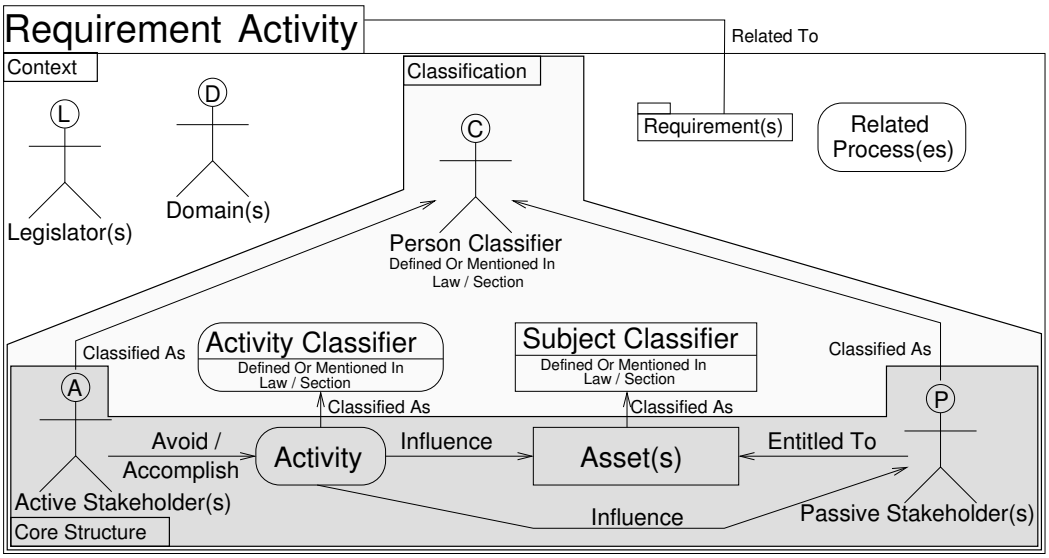


Fig. B.9: Law Identification Pattern

Solution — **Structure**

To bridge the gap between different wordings and to facilitate the discussion between requirements engineers and legal experts, we derived a Law Identification Pattern to support identifying relevant laws based on functional requirements of a system-to-be. We especially use the knowledge about laws which is described using the Law Pattern and the simultaneously established hierarchies of terms explained in Section B.1.2 about the Law Pattern. Note that we strongly suggest to use these two patterns in combination, but one can use any other method to describe laws as long as this method explicitly collects and describes the important terms for a law. In the following, we also consider requirements as central source of knowledge about the system to be. But it is possible to use other sources as long as the elements of the Law Identification Pattern can be derived from them. Hence, the Law Identification Pattern is not only usable in the requirements engineering context, but we will focus on this field. In our experience, requirements are available for each system, but the form may vary.

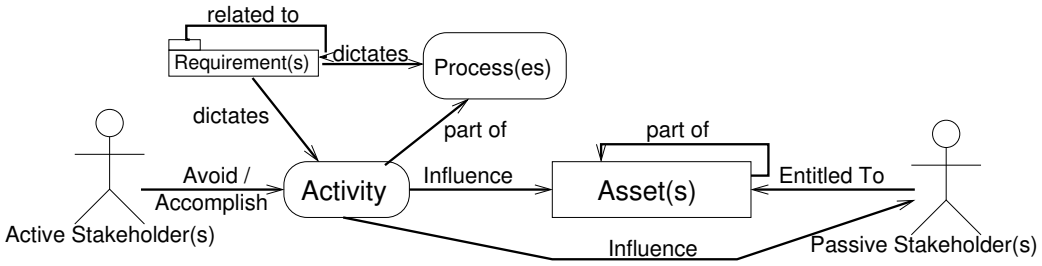


Fig. B.10: Relations between important elements which describe the system-to-be

Important for the matching with laws are the functional requirements, the activities the requirements contain, the assets which are important for the fulfillment of the requirements, and the relations between them and important domain knowledge. Figure B.10 shows these relations. First of all, a *Requirement* can be related to other *Requirements* and dictates a certain behavior of the machine. A behavior can be a certain *Activity* or a whole *Process*. A *Process* consists of different *Activities*. An *Activity* is avoided or accomplished by an *Active Stakeholder* and influences an *Asset*. Additionally, an *Activity* influences a *Passive Stakeholder* in a direct way or indirectly through an *Asset* to which the *Passive Stakeholder* is entitled. In addition, *Assets* can be related to each other, e.g. one *Asset* is part of another *Asset*. The documented information serves then as a basis for the law identification and is used for instantiating the Law Identification Pattern.

Entities — **Indirect Environment**

Context Contains all elements which are not of direct relevance for fulfilling the requirement described in the core structure, but which relate the specific requirement to its broader context.

Solution

Entities

Direct Environment

Core Structure Contains the elements of the requirement which describe one important core aspect of the requirement. These elements form the case description which is later on used for identifying relevant laws.

Constituent

Classification The classification part reflects the translation of terms specific for the system-to-be into legal terms.

Indirect Stakeholder

Legislator The legislator describes the law enacting force of a country in which jurisdiction the system-to-be will operate. Hence, these enacted and enforced laws might be of relevance for the system-to-be.

Domain The domain represents a specific domain, for example a certain business domain, to which the system-to-be will be related.

Direct Stakeholder

Active Stakeholder This is a stakeholder which directly interacts with the system, and who has to be enabled to do a certain activity or refrained from doing an certain activity. The active stakeholder is actively involved in executing the activity.

Passive Stakeholder This is a stakeholder who does not take active part in the activity, but who is influenced by this activity, or one asset which belongs to him / her is influenced.

Stakeholder

Person Classifier This is the legal term which can be used to classify a stakeholder of the core structure. Hence, in case of a legal revision, the system specific term used in the core structure for the stakeholder is replaced by this legal equivalent.

Activity(ies) describe(s) actions that an active stakeholder has to accomplish or avoid to fulfill the requirement.

Activity

Activity Classifier This is the legal term which can be used to classify an activity of the core structure. Hence, in case of a legal revision, the system specific term used in the core structure for the activity is replaced by this legal equivalent.

related Process(es) describe(s) a sequence of activities to accomplish a certain task. One of the activities of this process is the activity part of the core structure at hand. Implications for the activity might be of relevance for all other or some other activities of the process.

Solution

Entities

Resource

Asset describes some resource which is part of the system-to-be or its environment, which is influenced in some way by an activity.

Subject Classifier This is the legal term which can be used to classify an asset of the core structure. Hence, in case of a legal revision, the system specific term used in the core structure for the asset is replaced by this legal equivalent.

Requirement(s) One requirement is often part of a bigger set of requirements, which are closely related. For example, the fulfillment of one requirement is only necessary in case another requirement is fulfilled before.

Collaborations

Relation

Avoid / Accomplish Relates the active stakeholder with the activity(ies) he / she must do (accomplish) or must not do (avoid) to fulfill the requirement at hand.

Influence Relates activity(ies) with the assets or passive stakeholder which are influenced by the activity(ies).

Entitled To Relates active stakeholder(s) with asset(s). Such an relation exists whenever active stakeholder(s) have certain rights regarding the asset(s).

Classified As Relates system-specific terms with their legal equivalent.

Method

The process of instantiating Law Identification Patterns, as depicted in Fig. B.11, starts with a *context description* which we use to *identify legislator(s) and domain(s)*. The resulting *legislator(s)* are the ones relevant for the jurisdictions the system-to-be will operate in. The *domain(s)* are the ones for which the system-to-be is developed. In the next step, we select one *requirement* from the set of *requirements* which describe the system-to-be. Note that we only consider the functional requirements. Based on the selected requirement and any kind of *process description*, we *identify activity(ies) and process(es)* relevant for fulfilling the requirement. Note that it depends on the requirements formulation if all *activities* can be derived from the requirement, or if the process description is not only used for finding the according *processes* but also for completing the relevant activities. We *select* one *activity* from the *activities* and *identify the active stakeholders* for this activity using the according *requirement*. The resulting *active stakeholders* are the stakeholders necessary for executing the activity and which have an active influence on how the activity is executed. Next, we *identify the asset(s)* for the *activity* at hand and the according *requirement*. The resulting *asset(s)* are the resources which are influenced in some way when executing the activity. Using the identified *asset(s)*, we can *identify the passive stakeholders* for the *activity* and *requirement* at hand. The resulting *passive stakeholders* are stakeholders which are influenced by the *activity*, which is part of the *requirement* at hand, or who has a certain interest in one of the *assets*. Next, we have to *classify* the elements collected up to this point.

Solution

Method

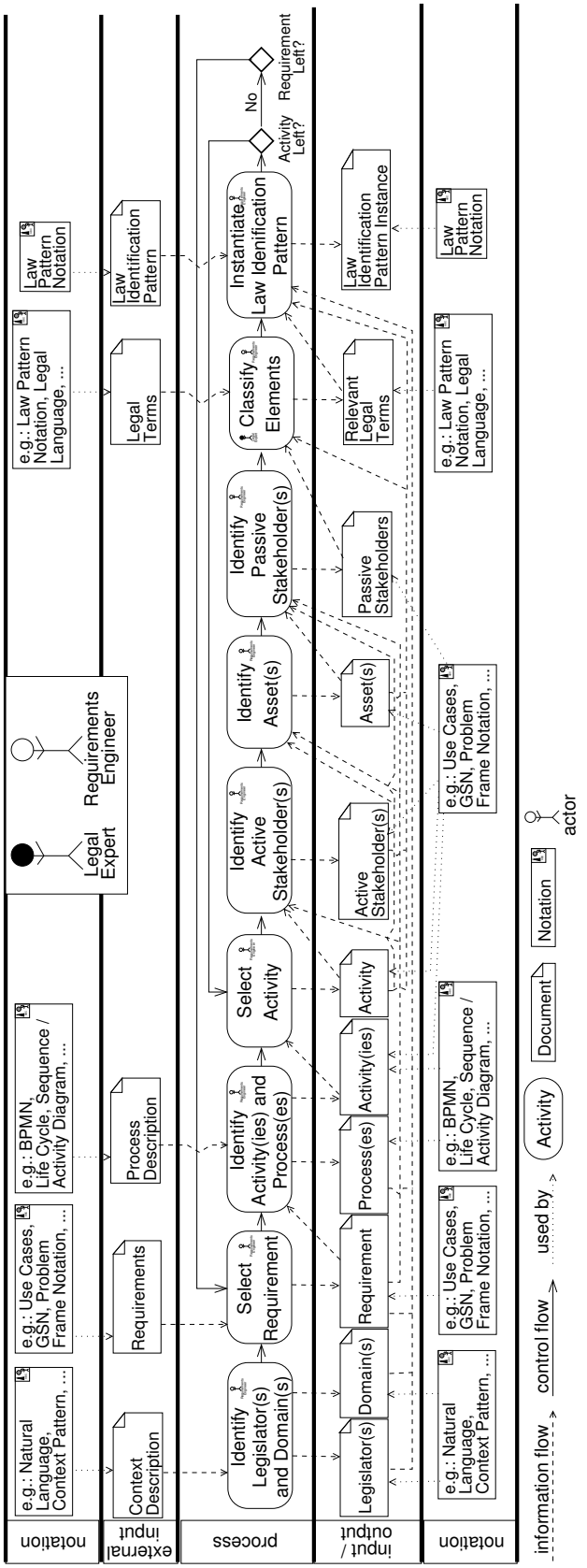


Fig. B.11: Law Identification Pattern Instantiation Process

***Solution* Method**

For executing this step, legal experts and requirements engineers have to collaborate to map *legal terms* to *active stakeholder(s)*, *passive stakeholder(s)*, *asset(s)* and the *activity* to get the *relevant legal terms*. All the information compiled up to this point is then used to *instantiate a Law Identification Pattern*. The *Law Identification Pattern instance* is the final result.

B.1.4. Smart Home Pattern

We describe in this section the results of our collaboration with domain experts from one of our NESSoS[21] industrial partners in the smart grid domain. While the Smart Grid Pattern was used and helpful when initiating the collaboration with the industrial partner for understanding and describing the general context of the system-to-be, it turned out that for the specific task not the full view on the smart grid domain was necessary. The reason was that the ongoing activities relevant for the NESSoS project were only considering the electricity and end consumer part of a smart grid. Especially this smart grid sub-domain got some attention by several countries. Beside this focus, it also became obvious that for continuous working with the Smart Grid Pattern in the context of this particular industry partner it was helpful to adapt the wording to the one used within the company. The result of focusing on the sub-domain of electricity and end-consumers, and reflecting the companies wording is the Smart Home Pattern.

One might ask if a company specific modification is still a pattern. First, we had several descriptions used by the industrial partner which were written independently for different projects. These documents were aggregated and generalized to modify the Smart Grid Pattern. Hence, we have our several occurrences, which makes the Smart Home Pattern at least a company specific pattern. Second, the company specific documents were also checked against the publicly available documents we used for the Smart Grid Pattern. As result, the changed elements are named in a company specific way, but the semantic is in line with documents of a more general relevance. Hence, the Smart Home Pattern is a general pattern with a company specific wording.

This section about the Smart Home Pattern differs from the other sections in this chapter as it does not only contain the Smart Home Pattern (Section B.1.4.1), but it also contains a discussion about the relation between the Smart Home Pattern and the Smart Grid Pattern (Section B.1.4.2[22]). Note that the Smart Home Pattern description embodies texts which are copied for the Smart Grid Pattern, as the Smart Home Pattern was derived from it.

B.1.4.1. The Pattern

	Pattern Name	Smart Home Pattern	**Classification**	Organizational & Technical
	Related Patterns	Cloud System Analysis, SOA Stakeholder, SOA Layer, Peer to Peer, Smart Grid		
Summary	**Intent**	This pattern can be used to elicit the context of an application or system which is part of smart homes which are part of an electricity smart grid. The documentation needs addressed by this pattern might be related to standard or legal compliance, and preparation of a requirements engineering process.		
	Essence	The smart home is a complex scenario and it is difficult to identify all relevant domain knowledge about it. The smart home contains several distributed stakeholders which makes communication and building a common understanding inevitable. Moreover, documenting the domain knowledge of smart home in a way that is easy to understand and does not lack any details is difficult, as well. *Read our graphical Smart Home Pattern and its templates to understand the essence of the smart home context. Afterward, instantiate this pattern for your particular scenario variation of the smart home scenario.*		
	Essence	*The solution helps you to elicit all essential information about the smart home scenario in a structured way and not to forget relevant parts. In addition, the instantiated patterns can be used as documentation as an initial input for, for example, requirements engineering, security and compliance analysis.*		

[21] `http://www.nessos-project.eu`

[22] Page 431

Summary	**Known Uses**	• CC protection profiles for Smart Meters [76, 77] • The documentation of the OpenMeter project [297] • The British Smart Grid implementation program [116, 115] • Several documents form the NESSoS industrial partners.
Motivation	**Example**	One application domain of the NESSoS project which we took part in is the smart grid. Our task within this project was to deliver solutions to analyze the smart grid regarding security, privacy and compliance, for example with standards, in the early phase of the software development life cycle. The NESSoS industrial partners provided the scenario descriptions. While we are experienced in requirements engineering, security, privacy and compliance in general, we did not have any further knowledge about smart grids. Hence, the results of applying our methods were unsatisfactory in the beginning. The main problem was that we did not speak the language of the industrial partner and we did not understand what we needed to know and how to describe this knowledge. As we also did not know what the important parts of a smart grid are, we could not ask the right questions. The result was a slow, trial and error process annoying both sides. We had to change something to cope with this situation. While the Smart Grid Pattern was used and helpful when initiating the collaboration with the industrial partner for understanding and describing the general context of the system-to-be, it turned out that for specific task not the full view on the smart grid domain was necessary. The reason was that the ongoing activities relevant for the NESSoS project were only considering the electricity and end consumer part of a smart grid. Especially this smart grid sub-domain got some attention by several countries. Beside this focus, it also became obvious that for continuous working with the smart grid pattern in the context of this particular industry partner it was helpful to adapt the wording to the one used within the company.
	Context	Taken from Beckers, Faßbender, Heisel, and Suppan (2014 [51]): "Deriving from the definitions of the European Commission (Commission of the European communities, 2011 [102]), the European Smart Grid Task Force[23], and the Office of Electricity Transmission and Distribution[24], the Smart Grid can be described as a large, flexible, self-monitoring, auto-balancing, and self-regulating electricity infrastructure which uses two-way digital communication to gather and respond on information in an automated manner in order to improve the efficiency, reliability (meaning safety and security), and sustainability of the production and distribution of energy. This new infrastructure will be able to efficiently integrate the behavior and actions of all users connected to it. This means generators, consumers, those that do both, and other third parties that provide services outside of energy generation."

[23] `http://ec.europa.eu/energy/gas_electricity/smartgrids/taskforce_en.htm` (last visited on 15-12-2013)
[24] `http://energy.gov/oe/technology-development/smart-grid` (last visited on 15-12-2013)

Problem

The general problem of describing the context within a smart grid is centered around different questions:

1. Which elements and information have to be collected to use the context description to judge compliance?
2. Which elements and information have to be collected to use the context description as an input for a basic security and privacy assessment?
3. How to collect knowledge about the important elements of a smart home in a structured way?
4. How to improve the communication process between different stakeholders?
5. How to externalize and store the collected information that it is useful afterward for the different addresses?

Motivation

Forces

There are several factors which have an influence on the context elicitation, making it complicated to find a solution:

Distributed Infrastructure The smart grid has a distributed nature. As a consequence, the information about the context for a system-to-be might not only reside with the stakeholder of the system-to-be, but other players in the smart grid which are not directly concerned with the system-to-be. Hence, cross-organizational knowledge has to be discovered.

Many and diverse standards and regulations The smart grid is a central building block for commodity distribution in the future for many countries, and incidents and issues with the smart grid have a severe impact on societies. Hence, many regulations and standards exist. To collect all these regulations and reflect their impact on the system-to-be is challenging.

Long living systems Smart grids are long living systems whose parts are difficult to evolve. Even though the grid itself is highly flexible regarding the part of taking elements and their interplay, the hardware used is planned to be operating for centuries. Hence, issues in already operating grid elements due to missing or wrong information is a significant threat.

No best practice available The topic of smart grid is relatively new. Currently, smart grids are only used on a small scale[25]. Hence, there is no best practice one can follow.

[25] E.g. http://www.fiercesmartgrid.com/story/honeywell-builds-smart-grid-success-england/2012-01-24

Table B.7 Information Collection Template for Direct Stakeholders

Name What is the name or identifier of the stakeholder?

Description Which important properties does the stakeholder have? What characterizes the stakeholder? What is its place in the environment?

Motivation Which objectives does the stakeholder follow? Why does the stakeholder influence the organization(s) / provider(s)?

Top Level Goals Which top level goals does the stakeholder have?
☐ **Adaptability** ☐ **Compliance** ☐ **Economy** ☐ **Efficiency**
☐ **Evolvability** ☐ **Learnability** ☐ **Maintainability** ☐ **Modularity**
☐ **Performance** ☐ **Portability** ☐ **Privacy** ☐ **Reliability** ☐ **Resilience**
☐ **Re-Usability** ☐ **Robustness** ☐ **Safety** ☐ **Scalability** ☐ **Security**
☐ **Testability** ☐ **Understandability** ☐ **Usability**

Kind

☐ **Specific** Is the stakeholder a real entity? Is the stakeholder not used to represent a group?
☐ **Representative** Is the stakeholder a real existing entity? Is this stakeholder used as proxy for a group of homogeneous stakeholders?
☐ **Group** Is the stakeholder not a real existing entity? Is this stakeholder used to describe for a group of homogeneous stakeholders?
☐ **Role** Can this stakeholder be shared through groups of heterogeneous stakeholders? Are there well-defined rights and permissions for this stakeholder?

Stakeholder Relations

To	Type	Description
To which target is the stakeholder related?	Which kind of relation?	Detailed Description of the relation.

Table B.8 Information Collection Template for Resources

Name What is the name or identifier of the resource?

Description Which important properties does the resource have? What characterizes the resource? What is its place in the environment?

Grid Element Relations

To	Type	Description
Which Grid Element is related?	Which kind of relation?	Detailed Description of the relation.

Solution **Structure**

Table B.9 Information Collection Template for Indirect Stakeholders

Name What is the name or identifier of the stakeholder?

Description Which important properties does the stakeholder have? What characterizes the stakeholder? What is its place in the environment?

Motivation Which objectives does the stakeholder follow? Why does the stakeholder influence the organization(s) / provider(s)?

Kind

☐ **Specific** Is the stakeholder a real entity? Is the stakeholder not used to represent a group?
☐ **Representative** Is the stakeholder a real existing entity? Is this stakeholder used as proxy for a group of homogeneous stakeholders?
☐ **Group** Is the stakeholder not a real existing entity? Is this stakeholder used to describe for a group of homogeneous stakeholders?

Influence

On	Description	Severity
Which direct stakeholder is influenced?	Which kind of influence? What kind of enforcement? What is the base for the influence?	What is the rating for the severity of the influence?

optional entries

Law candidates *(Legislator)* Which laws which might be of relevance for the actual grid (part) to be developed?

Domain-specific regulations *(Domain)* Which domain-specific regulations including, for example, standards and best practice do exist?

Solution

Structure

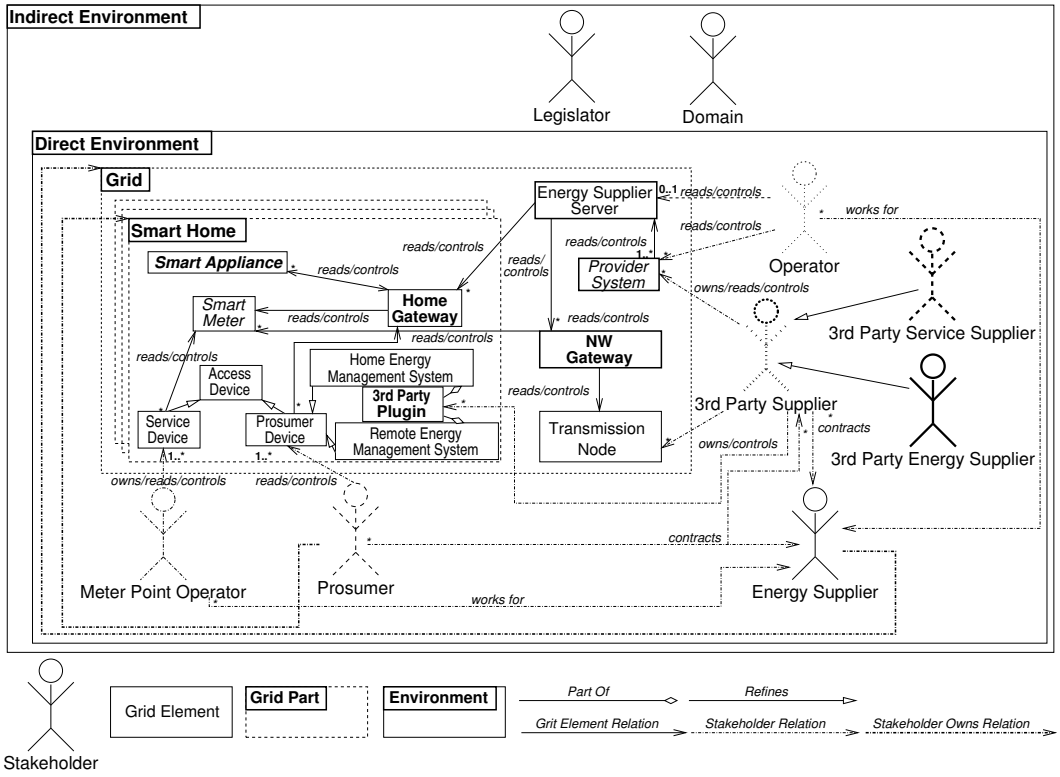


Fig. B.12: Important entities and relations for the Smart Home Pattern

Entities

Direct Environment Indirect Environment

Indirect Environment The indirect stakeholders as part of the indirect environment have no influence and, in most cases, also no interest in the machine itself. But they have an influence on the direct stakeholders and therefore they are important for the system-to-be.

Direct Environment The direct environment contains all the direct stakeholders, who have a direct relation to one or more parts of the grid or smart home. Hence, they are able to directly influence the grid or the smart home.

Solution Entities

Indirect Stakeholder

Legislator The legislator describes the government of a country, for example. A legislator enacts and enforces different regulations which the system-to-be has to be compliant to.

Domain The domain represents the special domains for which the system-to-be is developed. The domain's influence is based on the self regulations of a domain, standards for this domain and so forth. Note that the domain of a smart grid and smart home is given by default by choosing the pattern, but there might be more domains of relevance for a problem at hand.

Direct Stakeholder

Operator There are operators for different purposes, e.g., maintenance or billing. Operators work for the the energy supplier.

3rd Party Supplier 3rd party suppliers offer goods or services, which are delivered using the smart grid or are related to the smart home. The 3rd party providers also may have a contractual relation to the energy supplier.

3rd Party Service Supplier 3rd party service suppliers offer services, which are related to the smart home, for example, a smart appliance and the according 3rd party plugin.

3rd Party Energy Supplier 3rd party suppliers offer energy, which is delivered using the smart grid. The third party energy suppliers also have a contractual relation to the energy supplier.

Energy Supplier The energy supplier owns and operates the energy grid and its major parts. Note that a smart grid can be seen on different scales, starting from small-local ones to nationwide, to, for example, EU wide. Hence, there can be more than one energy supplier.

Prosumer The prosumers have a contractual relation with the energy supplier and consume energy which is provided by the energy supplier or third party energy suppliers. Note that this contractual relation can be transitive as a consumer might only have a direct contract with, for example, a third party energy supplier provider. Nevertheless, there are implicit contractual obligations between prosumers and energy suppliers. In addition, the prosumer can also sell small amounts of energy to the energy supplier and / or 3rd party energy supplier.

Meter Point Operator The meter point operators work for the energy supplier installing and maintaining the devices in the smart homes.

Solution Entities

Constituent

Grid The ICT-part of the grid is the thing to be built. This does not necessarily mean that all parts of the grid are object of a development project, but at least one will be the machine to be built.

Smart Home Part of the grid are the *smart homes* of concern. Note that a smart home represents the smallest accommodation unit, for example, a flat in apartment building but also a complete house in case the inhabitants form only one community.

Active Resource

Energy Supplier Server The central ICT element which coordinates and controls the grid.

Provider System Provider systems are (legacy) systems of an energy supplier or 3rd party supplier, which have no direct task in controlling the grid, but need access for different purposes, such as billing.

NW Gateway The network (NW) gateways are responsible for establishing the communication link to the smart meters. For long distance communication they use the power lines which they can access trough the transmission nodes.

Transmission Node A transmission node is used for realizing long distance power lines within a smart grid.

Home Gateway The device within the smart home which establishes the (W)LAN communication within the smart home. In most cases it is also connected to the Internet.

Smart Appliance Smart appliances are configurable devices such as heaters, which can be configured to turn on at a specific time or when certain conditions arise, for example, a certain temperature.

Smart Meter A smart meter is a means to measure and control the consumption of a certain commodity such as energy.

Access Device An access device is a means to control and communicate with smart meters.

Service Device An access device which used to maintain smart meters.

Prosumer Device An access device which is used by the prosumer for retrieving information from the smart meter and controlling the smart meter and different smart appliances.

Home Energy Management System A special prosumer device, which is always provided by the energy supplier to the prosumer, is the home energy management system, which allows, besides the viewing of energy values, the configuration of the smart appliances.

Remote Energy Management System Allows the access to the home energy management systems via remote devices such as smart phones.

Solution

Entities

Active Resource

3rd Party Plugins To extend the functionality of the remote / home energy management systems, the prosumer can buy 3rd party plugins from different 3rd party service providers. This can be simple GUI services for viewing information, but also complex new functionality, which e.g. requires a permanent Internet access to get information from the environment like weather data.

Collaborations

Relation

Stakeholder Relation Relates direct stakeholders and other elements of the smart grid. Three subtypes are possible:

Stakeholder to Grid Element The stakeholder is related to a smart grid element. The relation can be one of the following subtypes:

reads The stakeholder is able to retrieve information from a smart grid element.

controls The stakeholder is able to influence the smart grid element.

owns The stakeholder is the owner of the smart grid element

Stakeholder to Stakeholder The source stakeholder has some relation to the target stakeholder. We distinguish two relations:

works for The source stakeholder works for the target stakeholder.

contracts The source stakeholder has a contractual relation to the target stakeholder.

owns The source stakeholder owns all grid elements within the target area which do not have an explicit owner.

Grid Element Relation Relates elements of the grid. Four subtypes are distinguished:

reads The target grid element is read or provides information to the source grid element

controls The target grid element can be influenced to some extend by the source grid element. This includes also transitive control in the sense that the target grid element influences grid elements it controls on command of the source grid element.

refines The target element is refined by the source element. The source element adds new behavior or semantic.

part of The target element is part of the source element.

Method

The method starts with the *identification and selection* of the grid element which is the target of the context elicitation. In most cases, this is a single element. In rare cases, it is a set of grid elements or even a complete area. The selected elements have to be instantiated. Next, all *other grid elements* have to be *instantiated*. From our experience, a breadth-first traversal is reasonable. Normally, the elements closely related to the machine are well known and easy to instantiate, while far related elements are not that obvious and often only discovered when the close related elements are already instantiated. Note that in some cases not all grid elements are of relevance. Hence, they can be left out of the instantiated graphical pattern.

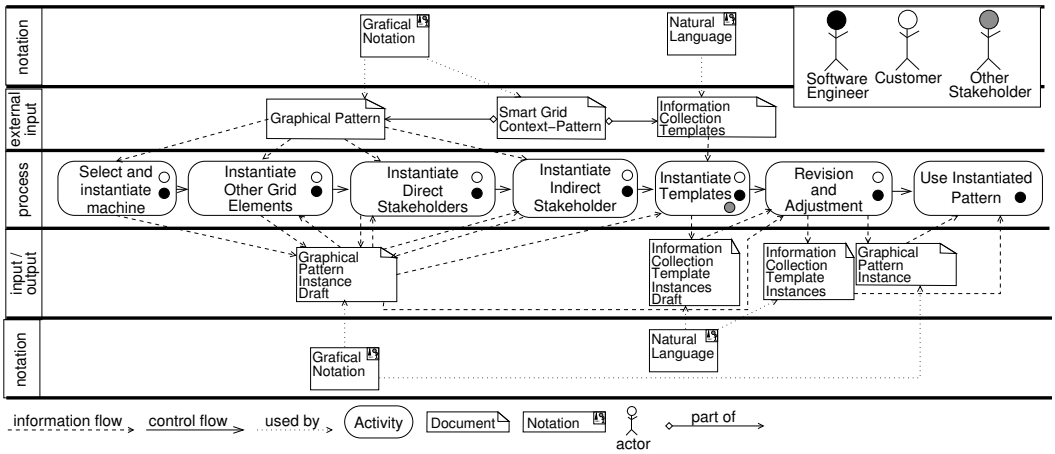


Fig. B.13: Process Pattern for Using the Smart Home Context Pattern to elicit the context.

Solution **Method**

When the grid elements are instantiated, one can start to *instantiate the direct stakeholders*. The fact that one already knows the important grid elements eases the instantiation of stakeholders. From our experience, when we start with the direct stakeholders, we often miss one, not having the grid element which is related to this stakeholder in mind. For *instantiating the indirect stakeholders*, one can iterate over the direct stakeholders and reason which domain and legislator are important for this direct stakeholder. When the graphical pattern is completely instantiated, all elements are described by *instantiating the information collection templates*. While for instantiation of the graphical pattern a small number of customer representatives are sufficient, the information collection involves more customer representatives. The reason is that in most cases the detailed information is scattered over different persons. Moreover, detailed information about important grid elements is not even available within the customers' organization. Hence, one must involve other, external stakeholders. The *final revision and adjustment* should be done with all customer representatives. The result is then *used* by the software engineers for their task. This activity is a complex one, which is a complete process in itself. For example, in Beckers, Faßbender, Heisel, and Suppan (2014 [51]) it is used for a threat analysis.

B.1.4.2. The Relation between the Smart Home Pattern and the Smart Grid Pattern

As already stated, the Smart Home Pattern is a focused and refined version of the Smart Grid Pattern. It focuses on the end-consumer and the electricity market. And it is aligned to a company specific wording and use.

Table B.10 gives an overview of the original entities of the Smart Grid Pattern (First column) and the related entities of the Smart Home Pattern (First row). An **x** in a table cell indicates that the entity of the Smart Home Pattern given by the row is related to the entity of the Smart Grid Pattern given by the column. The kind of relation is given by the second column or the second row, respectively. The *indirect environment* and the entities it contains are the same for both patterns. In the direct environment we have some changes. While the *operator* is the same entity in both patterns, the *3rd party provider* is renamed to *3rd party supplier* because of the company specific wording. Additionally, we added two specific kinds of 3rd party suppliers, namely the *3rd party service supplier* and the *3rd party energy supplier*. The reason for this addition was that these two specific kinds were of special interest in all smart home projects of

the company and therefore occurred in all documents. Hence, it is reasonable to make them explicitly visible. The *grid provider* was refined to the *energy supplier*, as the pattern is focused on the electricity market, and in this market a grid provider is always also an energy supplier. The *consumer* is refined to the *prosumer*. In the Smart Grid Pattern, the consumer represents all parties which consume energy. This includes companies and the like. In contrast, the prosumer only represents private customers living in a household. The *meter point operator* is a refinement of the *technician* in the Smart Grid Pattern.

One big difference between the Smart Home Pattern and the Smart Grid Pattern are the constituents *grid* and *micro grid* of the Smart Grid Pattern. For the smart home setting, the differentiation between grid and micro grid is not of relevance, because when analyzing a smart home setting, the technical details of the energy distribution are not the main concern. Hence, the grid and the micro grid are merged to one constituent representing the *grid*. In consequence, the entities which are part of the (micro) grid are also merged. The *grid controller* and the *micro grid controller* are merged and renamed to *energy supplier server*. The *grid sub controller* and the *micro grid sub controller* are merged and refined to the *NW (NetWork) Gateway*. Hence, all other functions a sub controller might have are removed from the Smart Home Pattern, as they describe technical details of the energy distribution which are not of relevance. The same applies for the *grid infrastructure device* and the *micro grid infrastructure device*. They are merged and refined to the *transmission node* only, as these nodes might also be used for data transmission. An entity which remains the same is the *provider system*.

The *micro grid* element, which can represent households, factories, and the like, is refined to the *smart home* as the whole pattern is focused on this particular element of a smart grid. One big change here is that for the electricity there is only one *smart meter* for a household. Hence, the *communication hub* and the *meters* are merged to one entity. The smart meter also only contains *actuators* and *sensors* for the main power line. All other meters with actuators and sensors are merged with the *other device* and renamed to the *smart appliances*. The reason is that in a smart home setting the electricity grid ends at the main power line to a household. Hence the grid is separated from the household internal electricity net by the smart meter. For the internal net, the prosumer can install any smart technical device he / she wants to use, but these devices are not regarded as part of the smart grid. In consequence, the smart meter has to be connected to the smart appliances. Therefore, the *home gateway* is added. The *access device*, *service device*, and the *consumer device* remain the same. The *consumer device* is only renamed to *prosumer device*. The *prosumer device* is further refined to the *(remote) home energy management system*, which is always part of a smart home. This central system is extensible by *3rd party plugins*, which are made explicitly visible.

The changed entities also have an impact on the relations. Table B.11[26] gives an overview of in which way a relation of the Smart Home Pattern (left hand side) was derived (center part) from the Smart Grid Pattern (right hand side). We will not go into detail on every relation, but focus on the ones which have not change by a simple merge or renaming.

One significant change is the relation between 3rd party supplier and 3rd party plugin (row 8). As a 3rd party supplier might provide 3rd party plugins the prosumer uses within his / her smart home, the 3rd party supplier has some kind of control over parts of the smart home. The reason is that nowadays such plugins often come in a form of apps or services which keep communicating with some infrastructure at the 3rd party suppliers side. Such an influence is not explicitly part of the Smart Grid pattern, as it is very specific for the smart home scenario and therefore not considered when describing smart grids in general.

Another significant change is due to the introduced home gateway. In the general smart grid setting it is assumed that the meters and other smart grid related devices within a micro grid element use some kind of infrastructure to communicate with each other. This infrastructure

[26] Page 435

Elements		Smart Grid Pattern																												
Smart Home Pattern	Mapping	Indirect Environment	Legislator	Domain	Direct Environment	Operator	3rd Party Supplier	3rd Party Service Supplier	3rd Party Energy Supplier	Energy Supplier	Prosumer	Meter Point Operator	Grid	Energy Supplier Server	Provider System	NW Gateway	Transmission Node	Smart Home	Smart Appliance	Home Gateway	Smart Meter	Access Device	Service Device	Prosumer Device	Home Energy Management System	3rd Party Plugins	Remote Energy Management System
Indirect Environment	Same	×																												
Legislator	Same		×																											
Domain	Same			×																										
Direct Environment	Same				×																									
Operator	Same					×																								
3rd Party Supplier	Same, but renamed						×																							
3rd Party Service Supplier	Refinement						×	×																						
3rd Party Energy Supplier	Refinement						×		×																					
Energy Supplier	Refinement									×																				
Prosumer	Refinement										×																			
Meter Point Operator	Refinement											×																		
Grid	Merged from different elements												×	×																
Energy Supplier Server	Refinement merged from different elements													×																
Provider System	Same														×															
NW Gateway	Refinement merged from different elements															×	×													
Transmission Node	Refinement merged from different elements																×													
Smart Home	Refinement																	×												
Smart Appliance	Merged from different elements																		×	×										
Home Gateway	New																													
Smart Meter	Merged from different elements																			×	×									
Access Device	Same																					×								
Service Device	Same																						×							
Prosumer Device	Same, but renamed																							×						
Home Energy Management System	Refinement																								×					
3rd Party Plugins	New																													
Remote Energy Management System	Refinement																													

Elements		Smart Grid Pattern																												
Smart Home Pattern	Mapping	Same	Same	Same	Same	Same	Same, but renamed	Refinement	Refinement	Refinement	Refinement	Refinement	Merged from different elements	Refinement merged from different elements	Same	Refinement merged from different elements	Refinement merged from different elements	Refinement	Merged from different elements	New	Merged from different elements	Same	Same	Same, but renamed	Refinement	New	Refinement
Indirect Environment	Same	×																												
Legislator	Same		×																											
Domain	Same			×																										
Direct Environment	Same				×																									
Operator	Same					×																								
3rd Party Supplier	Same, but renamed additionally refined to further elements						×																							
3rd Party Service Supplier	Refinement						×	×																						
3rd Party Energy Supplier	Refinement						×		×																					
Energy Supplier	Refinement									×																				
Prosumer	Refinement										×																			
Meter Point Operator	Refinement											×																		
Grid	Merged to another element												×																	
Grid Controller	Merged and refined to different elements													×																
Micro Grid	Merged to another element														×															
Micro Grid Controller	Merged and refined to different elements															×														
Provider System	Same																×													
Micro Grid Controller	Merged and refined to different elements																	×												
Micro Grid Sub Controller	Merged and refined to different elements																		×											
Grid Infrastructure Device	Merged and refined to another element																			×										
Micro Grid Infrastructure Device	Merged and refined to another element																				×									
Micro Grid Element	Refined																					×								
Actuator	Split up, merged and renamed																						×							
Sensor	Split up, merged and renamed																							×						
Other Devices	Merged and renamed																								×					
Meter	Split up and merged																									×				
Communication Hub	Merged and renamed																										×			
Access Device	Same																											×		
Service Device	Same																												×	
Consumer Device	Same, but renamed and refined to further elements																													×

Table B.10.: *Overview of Relations between Smart Home Pattern Entities and Smart Grid Pattern Entities*

can be independent from any other communication infrastructure within the micro grid element, for example, a ZigBee network or alike. In case of a smart home, an existing communication infrastructure is reused, namely a WLAN that already exists in the smart home. Hence, the smart appliances communicate with the home gateway (row 29) to access the smart meter (row 31). To ensure, for example, security properties, the energy supplier server might be able to access the home gateway (row 20). Note that the access from the energy supplier server to the home gateway is not used to access the smart meter. The direct access from the grid to the smart meter is established using an independent connection between the nw gateway to the smart meter (row 25).

The last significant change is concerned with the access device. The ability to connect to a smart meter directly is moved to the service devices only (row 34). Hence, the technician can access a smart meter directly, but the prosumer cannot access it directly. The prosumer devices connections rely on the home gateway (row 36) to access the smart meter (row 31) or the smart appliances (row 30).

Row	Smart Home Pattern Relation				Derived From	Smart Grid Pattern Relation			
	Type	Sub-Type	Source Element	Target Element	Mapping	Type	Sub-Types	Source Element	Target Element
1	Stakeholder Relation	reads / controls	Operator	Energy Supplier Server	Merged and refined target	Stakeholder Relation	reads / controls	Operator	Grid Controller
2					Merged and refined target	Stakeholder Relation	reads / controls	Operator	Micro Grid Controller
3	Stakeholder Relation	reads / controls	Operator	Provider System	Same	Stakeholder Relation	reads / controls	Operator	Provider System
4	Stakeholder Relation	works for	Operator	Energy Supplier	Renamed and refined target	Stakeholder Relation	works for	Operator	Grid Provider
5	Stakeholder Relation	owns / reads / controls	3rd Party Supplier	Provider System	Renamed source	Stakeholder Relation	owns / reads / controls	3rd Party Provider	Provider System
6	Stakeholder Relation	owns / reads / controls	3rd Party Supplier	Transmission Node	Renamed Source, and merge and refined target	Stakeholder Relation	owns / reads / controls	3rd Party Provider	Grid Infrastructure Device
7					Renamed Source, and merge and refined target	Stakeholder Relation	owns / reads / controls	3rd Party Provider	Micro Grid Infrastructure Device
8	Stakeholder Relation	owns / reads / controls	3rd Party Supplier	3rd Party Plugin	new				
9	Stakeholder Relation	contracts	3rd Party Supplier	Energy Supplier	Renamed source and refined target	Stakeholder Relation	contracts	3rd Party Provider	Grid Provider
10	Refines	-	3rd Party Service Supplier	3rd Party Supplier	new				
11	Refines	-	3rd Party Energy Supplier	3rd Party Supplier	new				
12	Stakeholder Owns Relation	-	Energy Supplier	Grid	Refined source and merged target	Stakeholder Owns Relation	-	Grid Provider	Grid
13					Refined source and merged target	Stakeholder Owns Relation	-	Grid Provider	Micro Grid
14	Stakeholder Relation	contracts	Prosumer	3rd Party Supplier	Refined source and renamed target	Stakeholder Relation	contracts	Consumer	3rd Party Provider
15	Stakeholder Relation	contracts	Prosumer	Energy Supplier	Refined source and target	Stakeholder Relation	contracts	Consumer	Grid Provider
16	Stakeholder Relation	reads / controls	Prosumer	Prosumer Device	Refined source and renamed target	Stakeholder Relation	reads / controls	Consumer	Consumer Device
17	Stakeholder Owns Relation	-	Prosumer	Smart Home	Refined source and target	Stakeholder Owns Relation	-	Consumer	Micro Grid Element
18	Stakeholder Relation	works for	Meter Point Operator	Energy Supplier	Refined source and target	Stakeholder Relation	works for	Technician	Grid Provider
19	Stakeholder Relation	reads / controls	Meter Point Operator	Service Device	Refined source and renamed target	Stakeholder Relation	reads / controls	Technician	Service Device
20	Grid Element Relation	reads / controls	Energy Supplier Server	Home Gateway	new				
21	Grid Element Relation	reads / controls	Energy Supplier Server	NW Gateway	Merged and refined source and target	Grid Element Relation	reads / controls	Grid Controller	Grid Sub Controller
22					Merged and refined source and target	Grid Element Relation	reads / controls	Micro Grid Controller	Micro Grid Sub Controller
23	Grid Element Relation	reads / controls	Provider System	Energy Supplier Server	Merged and refined target	Grid Element Relation	reads / controls	Provider System	Grid Controller
24	Grid Element Relation	reads / controls	NW Gateway	Smart Meter	Merged and refined source and target	Grid Element Relation	reads / controls	Micro Grid Controller	Communication Hub
25					Merged and refined source and target	Grid Element Relation	reads / controls	Micro Grid Sub Controller	Communication Hub
26					Merged and refined source and target	Grid Element Relation	reads / controls	Communication Hub	Meter
27	Grid Element Relation	reads / controls	NW Gateway	Transmission Node	Merged and refined source and target	Grid Element Relation	reads / controls	Grid Sub Controller	Grid Infrastructure Device
28					Merged and refined source and target	Grid Element Relation	reads / controls	Micro Grid Sub Controller	Micro Grid Infrastructure Device
29	Grid Element Relation	reads / controls	Smart Appliance	Home Gateway	new				
30	Grid Element Relation	reads / controls	Home Gateway	Smart Appliance	new				
31	Grid Element Relation	reads / controls	Home Gateway	Smart Meter	new				
32	Refines	-	Home Energy Management System	Prosumer Device	new				
33	Part Of	-	Home Energy Management System	3rd Party Plugin	new				
34	Grid Element Relation	reads / controls	Service Device	Smart Meter	new				
35	Refines	-	Service Device	Access Device	Same	Refines	-	Service Device	Access Device
36	Grid Element Relation	reads / controls	Prosumer Device	Home Gateway	new				
37	Refines	-	Prosumer Device	Access Device	Renamed Source	Refines	-	Consumer	Access Device
38	Refines	-	Remote Energy Management System	Prosumer Device	new				
39	Part Of	-	Remote Energy Management System	3rd Party Plugin	new				
40					removed	Stakeholder Relation	owns / reads / controls	3rd Party Provider	Access Device
41					removed	Stakeholder Relation	reads / controls	Consumer	Other Device
42					removed	Grid Element Relation	reads / controls	Grid Controller	Micro Grid Controller
43					removed	Grid Element Relation	reads / controls	Grid Sub Controller	Micro Grid Controller
44					removed	Refines	-	Micro Grid Controller	Micro Grid Sub Controller
45					removed	Grid Element Relation	controls	Meter	Actuator
46					removed	Grid Element Relation	reads	Meter	Sensor
47					removed	Refines	-	Communication Hub	Meter
48					removed	Grid Element Relation	reads / controls	Access Device	Meter
49					removed	Grid Element Relation	communicates	Other Device	Communication Hub
50					removed	Grid Element Relation	communicates	Communication Hub	Other Device

Table B.11.: *Overview of Relations between Smart Home Pattern Relations and Smart Grid Pattern Relations*

B.1.5. SOA Stakeholder Pattern

	Pattern Name	SOA Stakeholder Pattern	**Classification**	Organizational & Technical
	Related Patterns	**SOA Layer**, Cloud System Analysis, Peer to Peer		
Summary	**Intent**	This pattern can be used to elicit the context of an application or system which is part of or realized as Service Oriented Architecture.		
	Essence	A service oriented architecture follows the principle of re-use and encapsulations along business process. And it heavily relies on services which are selected and used at run-time. These services are often not under control of the users. As consequence, the environment of the system-to-be contains many, sometimes hidden, stakeholders and complex relations between them. And the system-to-be is distributed and relies on inhomogeneous technology elements. *The SOA Stakeholder Pattern helps to find all relevant stakeholders for a SOA-based system-to-be, and the relations between the stakeholders as well as the relations between stakeholders and technology elements of the SOA. The solution helps you to elicit all essential information about the SOA scenario in a structured way and not to forget relevant parts. In addition, the instantiated patterns can be used as documentation as an initial input for, for example, requirements engineering, security and compliance analysis.*		
	Known Uses	• IBM reference SOA architecture and method [20, 19] • SoaML [164] • Service-Oriented Computing: A Research Roadmap [306, 305] • A Pattern Language for Process Execution and Integration Design in Service-Oriented Architectures [182] • A pattern language for service-based platform integration and adaptation [246] • Patterns: Service-Oriented Architecture and Web Services [127]		
Motivation	**Example**	We had once the situation that a student of ours was also involved in a small size company providing tailored SOA based solutions for customer relationship and enterprise resource management. As it is often the case for start up companies, the development of the product was done in an unstructured, ad hoc manner. And they had chosen SOA for the benefits on the technology level, but were unaware of its implications. They also were missing an overview of stakeholders and their influence on the system. As the customer base was growing and also the usage of the old product, it became obvious to the small company that the established system was not maintainable on the long run as too many requirements were completely overlooked in the beginning. This led to the situation that they planned to replace the old product by a complete rebuild. But they were really in need of a means which helped them to discover all the stakeholders and their relations as well as the technology elements and their relations, and a structured method to integrate the insights in a rigid development method.		
	Context	For a SOA, the knowledge about the environment and the stakeholders is even more a key to success than for other architectures. Conventional applications are often built for a generic use case, which was obtained by generalizing a set of usage scenarios.		

Motivation

Context

To use such an application, the environment has to be adapted to a generic use case. Thus, it often happens that organizations are built around their systems. Changing an organization is costly, and processes that are built to meet IT requirements are often inefficient. In contrast, one major aim of SOA is to enable organizations to build IT systems, which follow their business processes. To reach this aim, it is necessary to be able to adapt a certain single scenario and consider its peculiarities. Hence, the scenario for which a SOA is built has to be described in detail.

Problem

The general problem of describing the context within a SOA is centered around different questions:

1. How to find all stakeholders which have a direct or indirect influence on the system-to-be?
2. How do the business processes look like and who are the process actors and organizations which have to be supported by system-to-be?
3. How to identify the relevant organizations for a SOA and the stakeholders representing them?
4. How to find the goals of stakeholders concerning the system-to-be?
5. Which elements and information have to be collected to use the context description to judge compliance?
6. How to collect knowledge about the important elements of a SOA in a structured way?
7. How to improve the communication process between the stakeholders?
8. How to externalize and store the collected information that it is useful afterward for the different addressees?

Forces

There are several factors which have an influence on the context elicitation for a making it complicated to find a solution:

Distributed System A SOA has a distributed nature. As a consequence, the information about the context for a system-to-be might not only reside with the stakeholders of the system-to-be, but other players in the SOA which are not directly concerned with the system-to-be. Hence, cross-organizational knowledge has to be discovered.

Dynamic nature of a SOA In a SOA services and the providers of these services can be exchanged easily. But due to these dynamics one must be aware of possible changes and their impact on the system-to-be beforehand and considering them accordingly in the development. Traceability between all abstraction levels is desired for impact analysis.

Only partial control over a SOA Third party services can be used within a SOA but the user gains no control over them. Hence, enforcement of desired properties is more a contractual problem than a development problem. But for certain properties, like privacy, one might not want to hand over the responsibility to someone else. This has to be reflected in the analysis of the system-to-be right from the start.

Motivation **Forces**

Rapidly evolving systems As a SOA is process centered it changes according to the processes. And processes tend to change frequently. Hence, it is a problem to be aware of process changes changing the SOA.

Solution **Structure**

Table B.13 Information Collection Template for Direct Stakeholders

Name What is the name or identifier of the stakeholder?

Description Which important properties does the stakeholder have? What characterizes the stakeholder? What is its place in the environment?

Motivation Which objectives does the stakeholder follow? Why does the stakeholder influence the organization(s) / provider(s)?

Top Level Goals Which top level goals does the stakeholder have?
☐ **Adaptability** ☐ **Compliance** ☐ **Economy** ☐ **Efficiency**
☐ **Evolvability** ☐ **Learnability** ☐ **Maintainability** ☐ **Modularity**
☐ **Performance** ☐ **Portability** ☐ **Privacy** ☐ **Reliability**
☐ **Resilience** ☐ **Re-Usability** ☐ **Robustness** ☐ **Safety** ☐ **Scalability**
☐ **Security** ☐ **Testability** ☐ **Understandability** ☐ **Usability**

Kind

☐ **Specific** Is the stakeholder a real entity? Is the stakeholder not used to represent a group?
☐ **Representative** Is the stakeholder a real existing entity? Is this stakeholder used as proxy for a group of homogeneous stakeholders?
☐ **Group** Is the stakeholder not a real existing entity? Is this stakeholder used to describe for a group of homogeneous stakeholders?
☐ **Role** Can this stakeholder be shared through groups of heterogeneous stakeholders? Are there well-defined rights and permissions for this stakeholder?

Stakeholder Relations

To	Type	Description
To which target is the stakeholder related?	Which kind of relation?	Detailed Description of the relation.

optional entries

Takes Part In *(Process Actor)*

Process	Activity
In which process does the actor participate?	Which activity does the actor enact?

Influence *(Process Actor)*

On	Description
Which other actor is influenced by the actor at hand?	Which kind of influence does the actor at hand have to the target actor?

Provides *(* Provider)*

Provides	Description
What is provided?	Whats the purpose of provide part?

Table B.14 Information Collection Template for Indirect Stakeholders

Name What is the name or identifier of the stakeholder?

Description Which important properties does the stakeholder have? What characterizes the stakeholder? What is its place in the environment?

Motivation Which objectives does the stakeholder follow? Why does the stakeholder influence the organization(s) / provider(s)?

Top Level Goals Which top level goals does the stakeholder have?
☐ **Adaptability** ☐ **Compliance** ☐ **Economy** ☐ **Efficiency**
☐ **Evolvability** ☐ **Learnability** ☐ **Maintainability** ☐ **Modularity**
☐ **Performance** ☐ **Portability** ☐ **Privacy** ☐ **Reliability**
☐ **Resilience** ☐ **Re-Usability** ☐ **Robustness** ☐ **Safety** ☐ **Scalability**
☐ **Security** ☐ **Testability** ☐ **Understandability** ☐ **Usability**

Kind

☐ **Specific** Is the stakeholder a real entity? Is the stakeholder not used to represent a group?
☐ **Representative** Is the stakeholder a real existing entity? Is this stakeholder used as proxy for a group of homogeneous stakeholders?
☐ **Group** Is the stakeholder not a real existing entity? Is this stakeholder used to describe for a group of homogeneous stakeholders?

Influence

On	Description	Severity
Which organization / provider is influenced?	Which kind of influence? What kind of enforcement? What is the base for the influence?	What is the rating for the severity of the influence?

Relation to other stakeholders

To	Description
Which other indirect stakeholder is related to the stakeholder at hand?	Which kind of relation?

optional entries

Law candidates *(Legislator)* Which laws which might be of relevance for the actual SOA to be developed?

Domain-specific regulations *(Domain)* Which domain-specific regulations including, for example, standards and best practice do exist?

Shares *(Shareholder)* Which kind of shares owns the shareholder? How many of them are property of the shareholder?

Assets *(Asset Provider)*

Asset	Description	Provided To
What are the assets owned by the asset provider?	What are the properties of the asset? How can it be characterized?	To which organization is the asset provided?

Structure

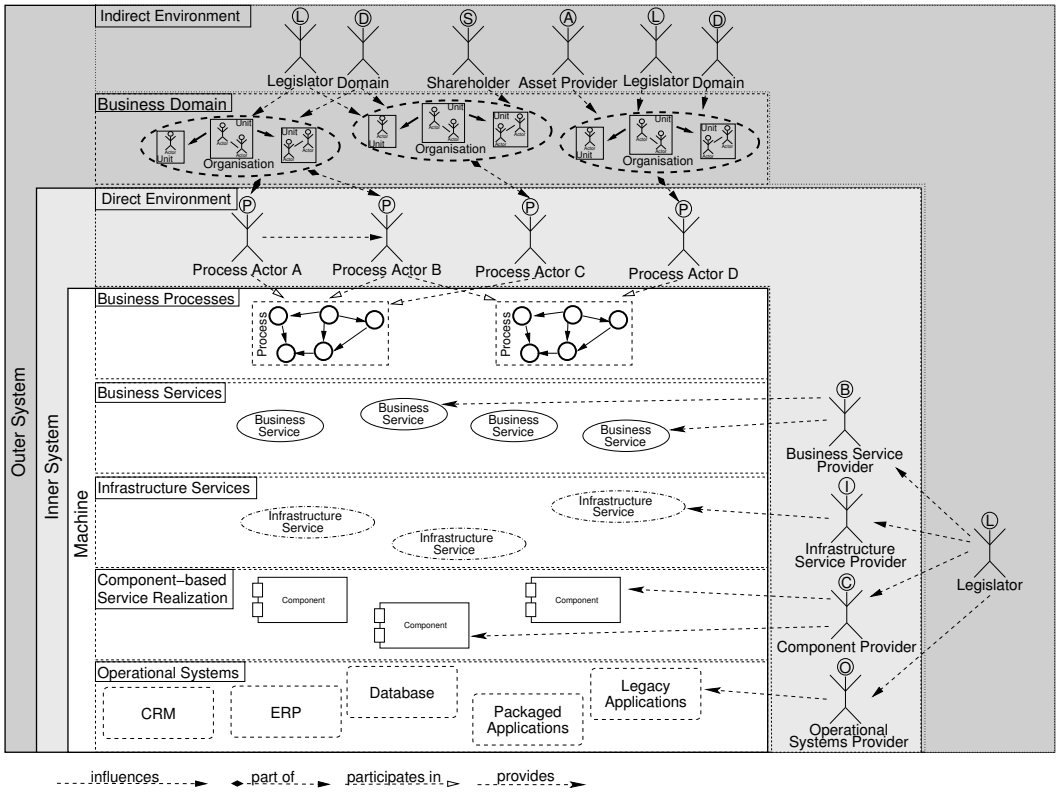


Fig. B.14: Important entities and relations for the SOA Stakeholder Pattern

Solution

Entities

Layer

Business Domain Layer which represents the real world in which the business supported by the SOA takes place.

Business Process Layer which contains the business processes established to run the business.

Business Service Layer which contains the services realizing business functionality.

Infrastructure Service Layer which contains the services realizing infrastructure functionality.

Component-Based Service Realization Layer which contains components used to realize services.

Operational Systems Layer which contains the operational systems needed for running the infrastructure.

Constituent

Inner System Spans the machine and its direct environment. All contained entities and information about them are needed to describe the behavior of a system-to-be.

Outer System Includes the inner system and adds the entities indirectly related to the system-to-be.

Solution Entities

Machine

Machine The machine area spans the SOA layers that form the machine. The business processes describe the behavior of the machine. The business services, infrastructure services, components, and operational systems describe the structure of the machine. Note that the business processes are not part of the machine altogether, as the processes also include actors, which are not part of the machine. Thus, the processes are the bridge between the SOA machine and its environment.

Indirect Environment

Indirect Environment It comprises all entities not related to the machine but to the direct environment. The business domain layer is one bridge between the direct and indirect environment. On the one hand, some entities of the direct environment are part of organizations. On the other hand, some entities of the indirect environment influence one or more organizations.

Direct Environment

Direct Environment The direct environment includes all entities, which participate in the business processes or provide a part, like a component, of the machine. An entity is something that exits in the environment independently of the machine or other entities.

Process

Process To run the business, certain processes are executed. Organizations participate in these processes. A process can be executed within one organization or across organizations' borders.

Indirect Stakeholder

Organizations The business domain consists of organizations, their structure and actors, and their business relations to each other.

Legislator The legislator describes the government of a country, for example. A legislator enacts and enforces different regulations which the system-to-be has to be compliant to.

Shareholder A shareholder brings in a certain asset and gets a share of the organization in exchange. In most cases the asset is money. The exchange implies that the asset is owned by the organization afterward. The share of the organization ensures a specified degree of influence for the shareholder. Shareholders are the primary source of business goals of an organization.

Asset Provider Asset providers cede a material or immaterial asset to one or more organizations. Unlike shareholders, they remain owners of this asset, and therefore a long-term contractual relation is established. Such a contract implies a certain influence on the organization.

Solution Entities

Indirect Stakeholder

Domain The domain represents the special domains for which the system-to-be is developed. The domain's influence is based on the self regulations of a domain, standards for this domain and so forth. Note that the domain of SOA is given by default by choosing the pattern, but there might be more domains of relevance for a problem at hand.

Process actors Process actors are part of an organization. A process actor can represent an entire organization, a role within this organization, or a specific person. This depends on the usage of the pattern and the level of detail needed when used. A process actor participates in one or more business processes. In some cases a process actor does not only influence other actors through the process, but also influences them directly. For example, one actor can be the supervisor of another actor.

Direct Stakeholder

Process actors Process actors are part of an organization. A process actor can represent an entire organization, a role within this organization, or a specific person. This depends on the usage of the pattern and the level of detail needed when used. A process actor participates in one or more business processes. In some cases a process actor does not only influence other actors through the process, but also influences them directly. For example, one actor can be the supervisor of another actor.

Business Service Provider A business service provider can be a representative for a whole group of providers, or be a specific one. Business service providers are selected at run-time, and in most cases business services offered by providers remain under their full control.

Infrastructure Service Providers An infrastructure service provider can be a representative for a whole group of providers, or be a specific one. Infrastructure service providers are selected at run-time, and in most cases infrastructure services offered by providers remain under their full control.

Component Provider A component provider can be a representative for a whole group of providers, or be a specific one. Infrastructure service providers are selected at implementation time, and in most cases components offered by component providers do not remain under their control.

Operational Systems Provider A operational systems provider can be a representative for a whole group of providers, or be a specific one. Operational systems providers are selected at implementation time, and in most cases components offered by operational systems providers do not remain under their control.

Solution **Entities**

Active Resource

Operational System Operational systems, like databases or legacy systems, are part of the last SOA layer at the bottom of the SOA stack.

Component Encapsulated piece of software which is not running by itself but can be integrated in another software to solve a problem.

Infrastructure Service All business services rely upon Infrastructure Services, which form the fourth layer. The infrastructure services offer the technical functions needed for the business services. These technical functions are implemented especially for the SOA or expose interfaces from the operational systems used in an organization.

Business Service Business processes are supported by Business Services, which form the the Business Service layer. A business service encapsulates a business function, which performs a process activity within a business process. Besides atomic business services, there are also composite business services, which rely on other business services. These services are built by composing other business services.

Solution **Collaborations**

Relation

Influence The source has some degree of influence on the behavior of the target.

Part Of The target is an organizational part of the source.

Participates In The source takes action in the target process.

Provides The source provides a part essential for the machine.

Solution **Method**

In the SOA Layer Pattern process (see Fig. B.15), we elicit the stakeholders of our SOA. Note that we assume that one used the SOA Layer pattern beforehand for the technical details. Therefore, we inspect each element of the *SOA Layer Pattern* instantiated previously. First, we instantiate the direct system environment (see Fig. B.14). We start with the organizations given in the *SOA Layer Pattern Instance*. For each process related to an organization, we identify the process actors, which act on behalf of the organization in this particular process. There has to be at least one process actor for each organization-process-relation. For all process actors, we have to instantiate the corresponding *Direct Stakeholder Templates*. Finally, we have to establish the relations between associated process actors.
Next, we inspect each business service, infrastructure service, component and operational system, whether there are already known provider(s) or not. When the providers are already known, we instantiate *Direct Stakeholder Template* for them and add them and the corresponding relations to the *SOA Stakeholder Pattern* instance. Further, we instantiate the indirect system environment. We also start with the organizations. We analyze for each organization at hand if there are relevant legislators, domains, shareholders and asset providers. For each identified indirect stakeholder, we instantiate the corresponding *Indirect Stakeholder Template*, and we add the indirect stakeholder and their relations to *SOA Stakeholder Pattern* instance. We repeat this procedure for all providers we find in the direct system environment.

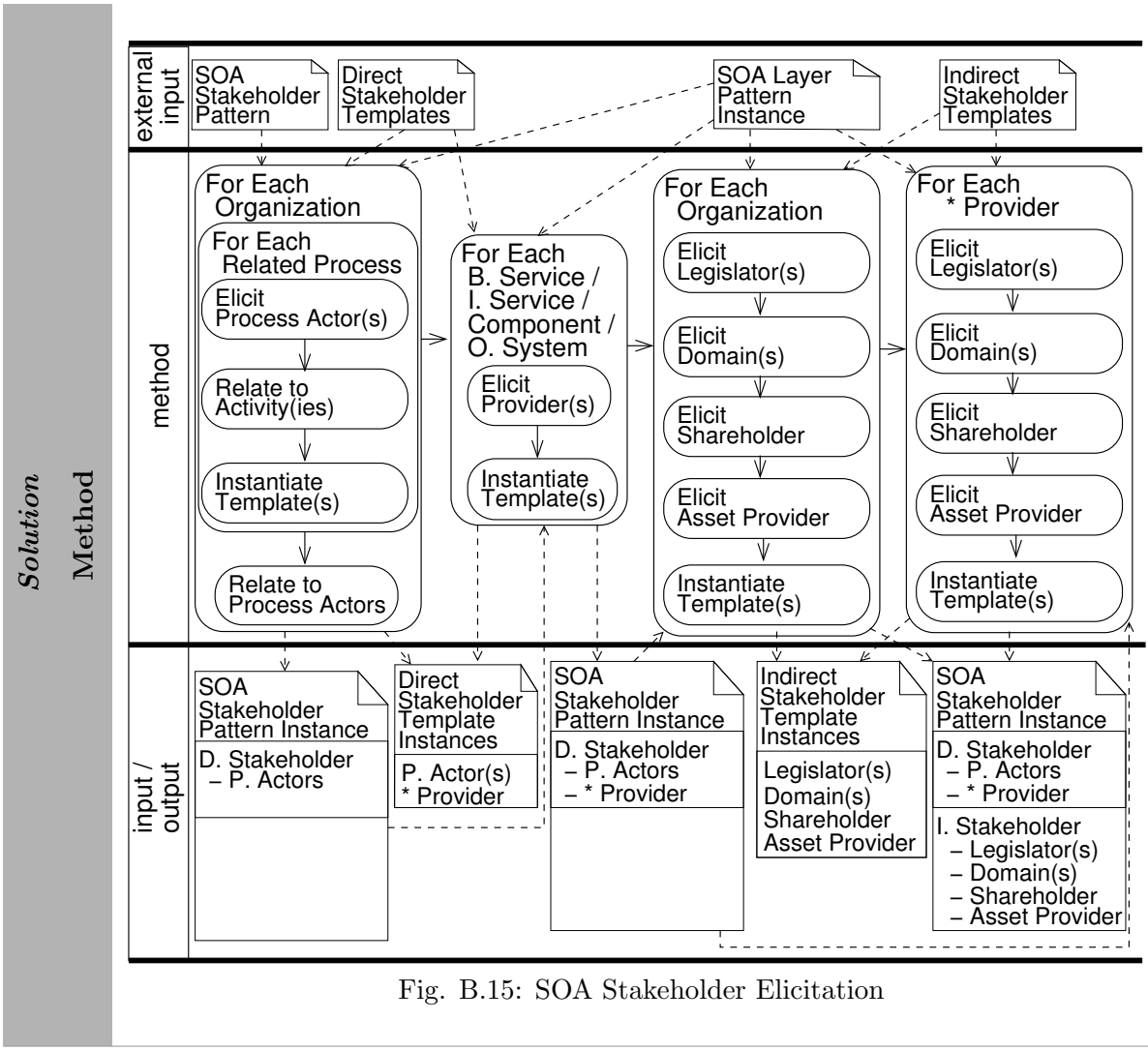


Fig. B.15: SOA Stakeholder Elicitation

B.2. Context Pattern Relations

Table B.15.: *Pattern Relation SOA Stakeholder Pattern to Smart Grid Pattern*

Direction	SOA Stakeholder Pattern to Smart Grid Pattern, Smart Grid Pattern to SOA Stakeholder Pattern
Relation Type	can refine
Reasoning	Some systems, such as a grid controller, of a Smart Grid can be realized using a SOA. Hence, the information in the SOA Stakeholder Pattern can be seen as a refinement of the context for these systems. But it is also possible that the SOA Stakeholder Pattern is used first to elicit information about a system-to-be which is part of a smart grid. Hence, extending the context information using the Smart Grid Pattern is reasonable.
Details	Table B.25 (Page 449)

Table B.16.: *Pattern Relation Cloud System Analysis Pattern to Smart Grid Pattern*

Direction	Cloud System Analysis Pattern to Smart Grid Pattern, Smart Grid Pattern to Cloud System Analysis Pattern
Relation Type	can refine
Reasoning	Some parts of a smart grid systems, such as a the data storage of a grid controller, can be realized using a cloud. Hence, the information in the Cloud System Analysis Pattern can be seen as a refinement of the context for these systems. But it is also possible that the Cloud System Analysis Pattern is used first to elicit information about a system-to-be which is part of a smart grid. Hence, extending the context information using the Smart Grid Pattern is reasonable.
Details	Not part of this thesis

Table B.17.: *Pattern Relation Cloud System Analysis Patter to Peer To Peer Pattern*

Direction	Peer to Peer Pattern to Cloud System Analysis Pattern
Relation Type	can refine
Reasoning	Specific technologies that form the core of the cloud, for example, the cloud database or the hypervisor, are likely based on P2P-architectures. In addition, the services deployed in the cloud can also be based on P2P-architectures. In both cases the Peer to Peer pattern can refine the description of these services or cloud components.
Details	Table B.28 (Page 465)

Table B.18.: *Pattern Relation Smart Grid Patter to Peer To Peer Pattern*

Direction	Peer to Peer Pattern to Smart Grid Pattern
Relation Type	can refine
Reasoning	When coming to a Smart Grid scenario with a large number of smart meters and devices, and / or with a highly dynamic join and leave behavior of the smart meters and devices, it might be necessary to use peer to peer technologies to cope with the resulting challenges. In both cases the Peer to Peer pattern can refine the description of the complete Smart Grid or the specific smart meter or device under consideration.
Details	Not part of this thesis

Table B.19.: *Pattern Relation SOA Stakeholder Pattern to SOA Layer Pattern*

Direction	SOA Stakeholder Pattern to SOA Layer Pattern, SOA Layer Pattern to SOA Stakeholder Pattern
Relation Type	can refine, used jointly
Reasoning	The SOA Stakeholder Pattern can be refined using the SOA Layer Pattern to get more information about the technical context. Contrariwise, the SOA Stakeholder Pattern can be used to provide the organizational context for a SOA Layer Pattern.
Details	Table B.30 (Page 466)

Table B.20.: *Pattern Relation SOA Stakeholder Pattern to Law Identification Pattern*

Direction	SOA Stakeholder Pattern to Law Identification Pattern
Relation Type	input
Reasoning	The SOA Stakeholder Pattern can be used as the input for identifying relevant laws for the SOA application at hand.
Details	Table B.32 (Page 467)

Table B.21.: *Pattern Relation SOA Stakeholder Pattern to Peer to Peer Pattern*

Direction	Peer to Peer Pattern to SOA Stakeholder Pattern
Relation Type	can refine
Reasoning	The services described in the SOA Stakeholder Pattern can rely on P2P-architectures. The Peer to Peer Pattern can be used to create a refined description of these services and also reason if these services can fulfill certain quality requirements, for example, security. The isolated analysis of services in a SOA can be helpful when services shall be evaluated for the question if they can fulfill certain requirements at all. Hence, the Peer to Peer pattern can help excluding certain services from the SOA Stakeholder Pattern.
Details	Table B.29 (Page 465)

Table B.22.: *Pattern Relation Cloud System Analysis Pattern to Law Identification Pattern*

Direction	Cloud System Analysis Pattern to Law Identification Pattern
Relation Type	input
Reasoning	The Cloud System Analysis Pattern elements can be used as the input for identifying relevant laws for the cloud at hand using the Law Identification Pattern.
Details	Table B.33 (Page 467)

Table B.23.: *Pattern Relation Law Pattern to Law Identification Pattern*

Direction	Law Pattern to Law Identification Pattern
Relation Type	input, used jointly
Reasoning	When instantiating the Law Pattern for a law one result are the legal terms important for this law. One step when instantiating a Law Identification Pattern instance is to relate the technical terms used in the system description to the corresponding legal terms. Hence, the terms derived when using the Law Pattern can be input for the term mapping in the Law Identification Pattern.
Details	Table B.31 (Page 466)

Table B.24.: *Pattern Relation Smart Grid Pattern to Law Identification Pattern*

Direction	Smart Grid Pattern to Law Identification Pattern
Relation Type	input
Reasoning	The Smart Grid Pattern elements can be used as the input for identifying relevant laws for the smart grid at hand using the Law Identification Pattern.
Details	

B.2.1. Smart Grid Pattern to SOA Layer Pattern

Row		Only in Smart Grid Pattern (SGP)	same	Mapping		Only in SOA Stakeholder Pattern (SSP)
				Element in SGP	Element in SSP	
1	Constituent	Grid, Micro Grid, Micro Grid Element				Inner System, Outer System
2	Machine			***Grid Controller***	**Machine**	
3				***Provider System***		
4				***Grid Sub Controller***		
5				***Micro Grid Controller***		
6				***Micro Grid Sub Controller***		
7				***Other Device***		
8	Direct Environment	Direct Environment				Direct Environment
9	Indirect Environment	Indirect Environment				Indirect Environment
10	Layer					Business Domain, Business Process, Business Service, Infrastructure Service, Component-Based Service Realization, Operational System
11	Process					Process
12	Relation	Stakeholder Relation (Stakeholder to Grid Element[reads, controls, owns], Stakeholder to Stakeholder[works for, contracts], owns), Grid Element Relation(reads, controls, refines, part of)				Influence, Part Of, Participates In, Provides
13	Indirect Stakeholder		Legislator, Domain	*3rd Party Provider*	Organizations	Shareholder, Asset Provider
14				*Grid Provider*		
15	Direct Stakeholder			Operator	Process Actors	Component Provider
16				Technician		
17				Consumer		
18				3rd Party Provider	Operational Systems Provider	
19				Grid Provider		
20				3rd Party Provider	Infrastructure Service Provider	
21				Grid Provider		
22				3rd Party Provider	Business Service Provider	
23				Grid Provider		
24				3rd Party Provider	*Organizations*	
25				Grid Provider		
26	Active Resource	Grid Infrastructure Device, Micro Grid Infrastructure Device, Actuator, Sensor		**Grid Controller**	***Machine***	Component, Infrastructure Service, Business Service
27				**Provider System**		
28				**Grid Sub Controller**		
29				**Micro Grid Controller**		
30				**Micro Grid Sub Controller**		
31				**Other Device**		
32				Grid Controller	**Operational System**	
33				Provider System		
34				Grid Sub Controller		
35				Micro Grid Controller		
36				Micro Grid Sub Controller		
37				Other Device		
38				Meter		
39				Communication Hub		
40				Access Device		
41				Service Device		
42				Consumer Device		

Table B.25.: *Smart Grid Pattern to SOA Stakeholder Pattern Relation Investigation Table*

Table B.26.: *Pattern Relation Reasoning Forms for Smart Grid Pattern and SOA Stakeholder Pattern*

Relation
Form 1
Source Element Grid Controller **Target Element** Machine **Relation Type** ☐ Same ☒ Mapped **Refinement** ☒ Source Refined ☒ Target Refined **Reliability** ☐ Unreliable ☒ Reliable **Reasoning** A serious problem of the IT infrastructure for a smart grid are the distributed nature and the interplay between different smart grid participants. The participants can change every time, for example a new 3rd party grid provider joins the market, so can the processes which are related to the smart grid. Hence, there are some efforts to tackle this problem by using a SOA for the smart grid IT infrastructure. As result, a grid controller might be (partially) designed using a SOA and services. If one starts in the smart grid context and the grid controller is the central element of concern and therefore the system-to-be, he/she might want to explore its SOA context. Contrariwise, if someone who is developing a SOA application and it turns out that this application is later used to control a smart grid, he/she might gain additional useful insights by instantiating the Smart Grid Pattern. **Source** ☐ Thought Experiment ☐ Own Experience ☒ Expert Experience ☒ Documentation **Example(s)** One example is the Spectrum Power SOA offered by Siemens (2015 [348]). There are also some scientific efforts to combine a smart grid with SOA, for example the work of Yang et al. (2011 [394]).
Form 2
Source Element Grid Sub Controller **Target Element** Machine **Relation Type** ☐ Same ☒ Mapped **Refinement** ☒ Source Refined ☒ Target Refined **Reliability** ☐ Unreliable ☒ Reliable **Reasoning** A serious problem of the IT infrastructure for a smart grid are the distributed nature and the interplay between different smart grid participants. The participants can change every time, for example a new 3rd party grid provider joins the market, so can the processes which are related to the smart grid. Hence, there are some efforts to tackle this problem by using a SOA for the smart grid IT infrastructure. As result, a grid controller might be (partially) designed using a SOA and services. If one starts in the smart grid context and the grid sub controller is the central element of concern and therefore the system-to-be, he/she might want to explore its SOA context. Contrariwise, if someone who is developing a SOA application and it turns out that this application is later used to control a smart grid, he/she might gain additional useful insights by instantiating the Smart Grid Pattern. **Source** ☐ Thought Experiment ☐ Own Experience ☒ Expert Experience ☒ Documentation **Example(s)** One example is the Spectrum Power SOA offered by Siemens (2015 [348]). There are also some scientific efforts to combine a smart grid with SOA, for example the work of Yang et al. (2011 [394]).

Continued on next page

Table B.26 – *continued from previous page*

Relation
Form 3
Source Element Micro Grid Controller **Target Element** Machine **Relation Type** ☐ Same ☒ Mapped **Refinement** ☒ Source Refined ☒ Target Refined **Reliability** ☐ Unreliable ☒ Reliable **Reasoning** A serious problem of the IT infrastructure for a smart grid are the distributed nature and the interplay between different smart grid participants. The participants can change every time, for example a new 3rd party grid provider joins the market, so can the processes which are related to the smart grid. Hence, there are some efforts to tackle this problem by using a SOA for the smart grid IT infrastructure. As result, a grid controller might be (partially) designed using a SOA and services. If one starts in the smart grid context and the micro grid controller is the central element of concern and therefore the system-to-be, he/she might want to explore its SOA context. Contrariwise, if someone who is developing a SOA application and it turns out that this application is later used to control a smart grid, he/she might gain additional useful insights by instantiating the Smart Grid Pattern. **Source** ☐ Thought Experiment ☐ Own Experience ☒ Expert Experience ☒ Documentation **Example(s)** One example is the Spectrum Power SOA offered by Siemens (2015 [348]). There are also some scientific efforts to combine a smart grid with SOA, for example the work of Yang et al. (2011 [394]).
Form 4
Source Element Grid Sub Controller **Target Element** Machine **Relation Type** ☐ Same ☒ Mapped **Refinement** ☒ Source Refined ☒ Target Refined **Reliability** ☐ Unreliable ☒ Reliable **Reasoning** A serious problem of the IT infrastructure for a smart grid are the distributed nature and the interplay between different smart grid participants. The participants can change every time, for example a new 3rd party grid provider joins the market, so can the processes which are related to the smart grid. Hence, there are some efforts to tackle this problem by using a SOA for the smart grid IT infrastructure. As result, a grid controller might be (partially) designed using a SOA and services. If one starts in the smart grid context and the micro grid sub controller is the central element of concern and therefore the system-to-be, he/she might want to explore its SOA context. Contrariwise, if someone who is developing a SOA application and it turns out that this application is later used to control a smart grid, he/she might gain additional useful insights by instantiating the Smart Grid Pattern. **Source** ☐ Thought Experiment ☐ Own Experience ☒ Expert Experience ☒ Documentation **Example(s)** One example is the Spectrum Power SOA offered by Siemens (2015 [348]). There are also some scientific efforts to combine a smart grid with SOA, for example the work of Yang et al. (2011 [394]).

Continued on next page

Table B.26 – *continued from previous page*

Relation
Form 5
Source Element Provider System **Target Element** Machine **Relation Type** ☐ Same ☒ Mapped **Refinement** ☒ Source Refined ☒ Target Refined **Reliability** ☐ Unreliable ☒ Reliable **Reasoning** A provider system can be any legacy system or new developed system which is connected to a grid controller but not concerned with the grid itself. And such a system can be realized using a SOA. When starting in the smart grid context and the system-to-be is a provider system which is / shall be realized as SOA, one might also use the SOA Stakeholder Pattern to gain further information. Contrariwise, when developing a SOA application which might be connected to a smart grid as provider system, it might be useful to know the smart grid context. **Source** ☐ Thought Experiment ☒ Own Experience ☒ Expert Experience ☒ Documentation **Example(s)** One example is the SAP NetWeaver technology offered by SAP (2015 [336]). It is used to connect SAP and other application using a SOA. The billing scenario within the NESSoS project is one example for such a usage.
Form 6
Source Element Other Devices **Target Element** Machine **Relation Type** ☐ Same ☒ Mapped **Refinement** ☒ Source Refined ☒ Target Refined **Reliability** ☐ Unreliable ☒ Reliable **Reasoning** A other device can be any legacy system or new developed system which is connected to a communication hub within a micro grid element. And such a system can be realized using a SOA. When starting in the smart grid context and the system-to-be is a other device which is / shall be realized as SOA, one might also use the SOA Stakeholder Pattern to gain further information. Contrariwise, when developing a SOA application which might be connected to a communication hub, it might be useful to know the smart grid context. **Source** ☒ Thought Experiment ☐ Own Experience ☒ Expert Experience ☒ Documentation **Example(s)** The idea of applications and devices which are integrated with a micro grid element using technologies such as clouds and SOA is not new. For example, IBM advertises solutions for such scenarios for years (IBM, 2010 [191]).
Form 7
Source Element Legislator **Target Element** Legislator **Relation Type** ☒ Same ☐ Mapped **Refinement** ☐ Source Refined ☐ Target Refined **Reliability** ☐ Unreliable ☒ Reliable **Reasoning** Of course a legislator remains the same in a smart grid and SOA context. **Source** ☐ Thought Experiment ☒ Own Experience ☒ Expert Experience ☐ Documentation **Example(s)** –

Continued on next page

Table B.26 – *continued from previous page*

Relation
Form 8
Source Element Domain **Target Element** Domain **Relation Type** ☒ Same ☐ Mapped **Refinement** ☐ Source Refined ☐ Target Refined **Reliability** ☐ Unreliable ☒ Reliable **Reasoning** Of course a domain remains the same in a smart grid and SOA context. **Source** ☐ Thought Experiment ☒ Own Experience ☒ Expert Experience ☐ Documentation **Example(s)** –
Form 9
Source Element 3rd Party Provider **Target Element** Organizations **Relation Type** ☐ Same ☒ Mapped **Refinement** ☐ Source Refined ☐ Target Refined **Reliability** ☐ Unreliable ☒ Reliable **Reasoning** In case a 3rd party provider is directly involved in the process(es) which are supported by the system-to-be, the provider might be one of the involved organizations in the SOA view. Contrariwise, an organization which is involved in the process(es) supported by an SOA application might be mapped to a 3rd party provider. Note that there might be organizations in the SOA view, which cannot be mapped to any Smart Grid Pattern element, as they are are not concerned with the smart grid at all. The other way round, a 3rd party provider might not be mapped at all, as the provider might not be related to the system-to-be build as SOA. **Source** ☐ Thought Experiment ☒ Own Experience ☒ Expert Experience ☒ Documentation **Example(s)** The billing process covers such a case, where an organization is involved in the process and also concerned with the smart grid itself. And an organization which is interested in billing might be a 3rd party provider.
Form 10
Source Element Grid Provider **Target Element** Organizations **Relation Type** ☐ Same ☒ Mapped **Refinement** ☐ Source Refined ☐ Target Refined **Reliability** ☐ Unreliable ☒ Reliable **Reasoning** In case a grid provider is directly involved in the process(es) which are supported by the system-to-be, the provider might be one of the involved organizations in the SOA view. Contrariwise, an organization which is involved in the process(es) supported by an SOA application might be mapped to a grid provider. Note that there might be organizations in the SOA view, which cannot be mapped to any Smart Grid Pattern element, as they are are not concerned with the smart grid at all. The other way round, a grid provider might not be mapped at all, as the provider might not be related to the system-to-be build as SOA. **Source** ☐ Thought Experiment ☒ Own Experience ☒ Expert Experience ☒ Documentation **Example(s)** The billing process covers such a case, where an organization is involved in the process and also concerned with the smart grid itself. And an organization which is interested in billing might be a grid provider.

Continued on next page

Table B.26 – *continued from previous page*

Relation
Form 11
Source Element Operator **Target Element** Process Actor **Relation Type** ☐ Same ☒ Mapped **Refinement** ☐ Source Refined ☐ Target Refined **Reliability** ☐ Unreliable ☒ Reliable **Reasoning** In case an operator is directly involved in the process(es) which are supported by the system-to-be, the operator is one of the involved process actors in the SOA view. Contrariwise, a process actor which is involved in the process(es) supported by an SOA application might be mapped to an operator. Note that there might be process actors in the SOA view, which cannot be mapped to any Smart Grid Pattern element, as they are are not concerned with the smart grid at all. The other way round, an operator might not be mapped at all, as the operator might not be related to the system-to-be build as SOA. **Source** ☐ Thought Experiment ☒ Own Experience ☒ Expert Experience ☒ Documentation **Example(s)** The billing process covers such a case, where a process actor is involved in the process and also concerned with the smart grid itself. And a process actor which is involved in the billing process might be an operator.
Form 12
Source Element Technician **Target Element** Process Actor **Relation Type** ☐ Same ☒ Mapped **Refinement** ☐ Source Refined ☐ Target Refined **Reliability** ☐ Unreliable ☒ Reliable **Reasoning** In case a technician is directly involved in the process(es) which are supported by the system-to-be, the technician is one of the involved process actors in the SOA view. Contrariwise, a process actor which is involved in the process(es) supported by an SOA application might be mapped to an technician. Note that there might be process actors in the SOA view, which cannot be mapped to any Smart Grid Pattern element, as they are are not concerned with the smart grid at all. The other way round, an technician might not be mapped at all, as the technician might not be related to the system-to-be build as SOA. **Source** ☒ Thought Experiment ☐ Own Experience ☒ Expert Experience ☐ Documentation **Example(s)** –
Form 13

Continued on next page

Table B.26 – *continued from previous page*

Relation
Source Element Consumer **Target Element** Process Actor **Relation Type** ☐ Same ☒ Mapped **Refinement** ☐ Source Refined ☐ Target Refined **Reliability** ☐ Unreliable ☒ Reliable **Reasoning** In case a consumer is directly involved in the process(es) which are supported by the system-to-be, the consumer is one of the involved process actors in the SOA view. Contrariwise, a process actor which is involved in the process(es) supported by an SOA application might be mapped to a consumer. Note that there might be process actors in the SOA view, which cannot be mapped to any Smart Grid Pattern element, as they are are not concerned with the smart grid at all. The other way round, an consumer might not be mapped at all, as the consumer might not be related to the system-to-be build as SOA. **Source** ☐ Thought Experiment ☒ Own Experience ☒ Expert Experience ☒ Documentation **Example(s)** The billing process covers such a case, where an process actor is involved in the process and also concerned with the smart grid itself. And a process actor which is involved in the billing process might be an consumer.
Form 14
Source Element 3rd Party Provider **Target Element** Operational Systems Provider **Relation Type** ☐ Same ☒ Mapped **Refinement** ☐ Source Refined ☐ Target Refined **Reliability** ☐ Unreliable ☒ Reliable **Reasoning** In case a 3rd party provider is not directly involved in the process(es) which are supported by the system-to-be, but one of the systems the provider controls is used as operational system for the system-to-be, the provider might be mapped to one of the involved operational systems provider in the SOA view. Contrariwise, an operational systems provider, who not only provides an operational system but who is also directly concerned with the smart grid itself by, for example, providing energy, might be mapped to a 3rd party provider. Note that there might be operational systems providers in the SOA view, which cannot be mapped to any Smart Grid Pattern element, as they are are not concerned with the smart grid at all. The other way round, a 3rd party provider might not be mapped at all, as the provider might not be related to the system-to-be build as SOA. **Source** ☒Thought Experiment ☐ Own Experience ☒ Expert Experience ☐ Documentation **Example(s)** –
Form 15

Continued on next page

Table B.26 – *continued from previous page*

Relation
Source Element Grid Provider **Target Element** Operational Systems Provider **Relation Type** ☐ Same ☒ Mapped **Refinement** ☐ Source Refined ☐ Target Refined **Reliability** ☐ Unreliable ☒ Reliable **Reasoning** In case a grid provider is not directly involved in the process(es) which are supported by the system-to-be, but one of the systems the provider controls is used as operational system for the system-to-be, the provider might be mapped to one of the involved operational systems provider in the SOA view. Contrariwise, an operational systems provider, who not only provides an operational system but who is also directly concerned with the smart grid itself by maintaining the grid infrastructure might be mapped to a grid provider. Note that there might be operational systems providers in the SOA view, which cannot be mapped to any Smart Grid Pattern element, as they are are not concerned with the smart grid at all. The other way round, a grid provider might not be mapped at all, as the provider might not be related to the system-to-be build as SOA. **Source** ☒Thought Experiment ☐ Own Experience ☒ Expert Experience ☐ Documentation **Example(s)** –
Form 16
Source Element 3rd Party Provider **Target Element** Infrastructure Service Provider **Relation Type** ☐ Same ☒ Mapped **Refinement** ☐ Source Refined ☐ Target Refined **Reliability** ☐ Unreliable ☒ Reliable **Reasoning** In case a 3rd party provider is not directly involved in the process(es) which are supported by the system-to-be, but one of the services the provider offers is used as infrastructure service for the system-to-be, the provider might be mapped to one of the involved infrastructure service provider in the SOA view. Contrariwise, an infrastructure service provider, who not only provides an infrastructure service but who is also directly concerned with the smart grid itself by, for example, providing energy, might be mapped to a 3rd party provider. Note that there might be infrastructure service providers in the SOA view, which cannot be mapped to any Smart Grid Pattern element, as they are are not concerned with the smart grid at all. The other way round, a 3rd party provider might not be mapped at all, as the provider might not be related to the system-to-be build as SOA. **Source** ☒Thought Experiment ☐ Own Experience ☒ Expert Experience ☒ Documentation **Example(s)** One example is the Spectrum Power SOA offered by Siemens (2015 [348]). The Spectrum Power SOA includes infrastructure services, such as an enterprise service bus or communication means, which can be can be replaced by infrastructure services offered by 3rd parties.
Form 17

Continued on next page

Table B.26 – *continued from previous page*

Relation
Source Element Grid Provider **Target Element** Infrastructure Service Provider **Relation Type** ☐ Same ☒ Mapped **Refinement** ☐ Source Refined ☐ Target Refined **Reliability** ☐ Unreliable ☒ Reliable **Reasoning** In case a grid provider is not directly involved in the process(es) which are supported by the system-to-be, but one of the services the provider provides is used as infrastructure service for the system-to-be, the provider might be mapped to one of the involved infrastructure service provider in the SOA view. Contrariwise, an infrastructure service provider, who not only provides an infrastructure service but who is also directly concerned with the smart grid itself by maintaining the grid infrastructure might be mapped to a grid provider. Note that there might be infrastructure service providers in the SOA view, which cannot be mapped to any Smart Grid Pattern element, as they are are not concerned with the smart grid at all. The other way round, a grid provider might not be mapped at all, as the provider might not be related to the system-to-be build as SOA. **Source** ☒Thought Experiment ☐ Own Experience ☒ Expert Experience ☒ Documentation **Example(s)** One example is the Spectrum Power SOA offered by Siemens (2015 [348]). The Spectrum Power SOA includes infrastructure services, such as an enterprise service bus or communication means, which can be invoked by 3rd parties.
Form 18
Source Element 3rd Party Provider **Target Element** Business Service Provider **Relation Type** ☐ Same ☒ Mapped **Refinement** ☐ Source Refined ☐ Target Refined **Reliability** ☐ Unreliable ☒ Reliable **Reasoning** In case a 3rd party provider is not directly involved in the process(es) which are supported by the system-to-be, but one of the services the provider offers is used as business service for the system-to-be, the provider might be mapped to one of the involved business service provider in the SOA view. Contrariwise, a business service provider, who not only provides a business service but who is also directly concerned with the smart grid itself by, for example, providing energy, might be mapped to a 3rd party provider. Note that there might be business service providers in the SOA view, which cannot be mapped to any Smart Grid Pattern element, as they are are not concerned with the smart grid at all. The other way round, a 3rd party provider might not be mapped at all, as the provider might not be related to the system-to-be build as SOA. **Source** ☒Thought Experiment ☐ Own Experience ☒ Expert Experience ☒ Documentation **Example(s)** One example is the Spectrum Power SOA offered by Siemens (2015 [348]). The Spectrum Power SOA includes business services, such as accounting, which can be can be replaced by business services offered by 3rd parties.
Form 19

Continued on next page

Table B.26 – *continued from previous page*

Relation
Source Element Grid Provider **Target Element** Business Service Provider **Relation Type** ☐ Same ☒ Mapped **Refinement** ☐ Source Refined ☐ Target Refined **Reliability** ☐ Unreliable ☒ Reliable **Reasoning** In case a grid provider is not directly involved in the process(es) which are supported by the system-to-be, but one of the services the provider provides is used as business service for the system-to-be, the provider might be mapped to one of the involved business service provider in the SOA view. Contrariwise, a business service provider, who not only provides a business service but who is also directly concerned with the smart grid itself by maintaining the grid infrastructure might be mapped to a grid provider. Note that there might be business service providers in the SOA view, which cannot be mapped to any Smart Grid Pattern element, as they are are not concerned with the smart grid at all. The other way round, a grid provider might not be mapped at all, as the provider might not be related to the system-to-be build as SOA. **Source** ☒Thought Experiment ☐ Own Experience ☒ Expert Experience ☒ Documentation **Example(s)** One example is the Spectrum Power SOA offered by Siemens (2015 [348]). The Spectrum Power SOA includes business services, such as accounting, which can be invoked by 3rd parties.
Form 20
Source Element Grid Controller **Target Element** Operational System **Relation Type** ☐ Same ☒ Mapped **Refinement** ☐ Source Refined ☒ Target Refined **Reliability** ☐ Unreliable ☒ Reliable **Reasoning** In case a grid controller is not the central system-to-be but is used as operational system for the system-to-be, the grid controller is mapped to an operational system in the SOA view. Contrariwise, an operational system, which is also directly integrated the the smart grid itself might be mapped to a grid controller. In case, the operational system plays a very central role in the SOA, it might be necessary to understand its smart grid context. Hence, it can be the reason for a refinement. Note that there might be operational systems in the SOA view, which cannot be mapped to any Smart Grid Pattern element, as they are are not concerned with the smart grid at all. The other way round, a grid controller might not be mapped at all, as it might not be related to the system-to-be build as SOA. **Source** ☒Thought Experiment ☐ Own Experience ☒ Expert Experience ☐ Documentation **Example(s)** –
Form 21

Continued on next page

Table B.26 – *continued from previous page*

Relation
Source Element Provider System **Target Element** Operational System **Relation Type** ☐ Same ☒ Mapped **Refinement** ☐ Source Refined ☒ Target Refined **Reliability** ☐ Unreliable ☒ Reliable **Reasoning** In case a provider system is not the central system-to-be but is used as operational system for the system-to-be, the provider system is mapped to an operational system in the SOA view. Contrariwise, an operational system, which is also directly integrated the the smart grid itself might be mapped to a provider system. In case, the operational system plays a very central role in the SOA, it might be necessary to understand its smart grid context. Hence, it can be the reason for a refinement. Note that there might be operational systems in the SOA view, which cannot be mapped to any Smart Grid Pattern element, as they are are not concerned with the smart grid at all. The other way round, a provider system might not be mapped at all, as it might not be related to the system-to-be build as SOA. **Source** ☒Thought Experiment ☐ Own Experience ☒ Expert Experience ☐ Documentation **Example(s)** –
Form 22
Source Element Grid Sub Controller **Target Element** Operational System **Relation Type** ☐ Same ☒ Mapped **Refinement** ☐ Source Refined ☒ Target Refined **Reliability** ☐ Unreliable ☒ Reliable **Reasoning** In case a grid sub controller is not the central system-to-be but is used as operational system for the system-to-be, the grid sub controller is mapped to an operational system in the SOA view. Contrariwise, an operational system, which is also directly integrated the the smart grid itself might be mapped to a grid sub controller. In case, the operational system plays a very central role in the SOA, it might be necessary to understand its smart grid context. Hence, it can be the reason for a refinement. Note that there might be operational systems in the SOA view, which cannot be mapped to any Smart Grid Pattern element, as they are are not concerned with the smart grid at all. The other way round, a grid sub controller might not be mapped at all, as it might not be related to the system-to-be build as SOA. **Source** ☒Thought Experiment ☐ Own Experience ☒ Expert Experience ☐ Documentation **Example(s)** –
Form 23

Continued on next page

Table B.26 – *continued from previous page*

Relation
Source Element Micro Grid Controller **Target Element** Operational System **Relation Type** ☐ Same ☒ Mapped **Refinement** ☐ Source Refined ☒ Target Refined **Reliability** ☐ Unreliable ☒ Reliable **Reasoning** In case a micro grid controller is not the central system-to-be but is used as operational system for the system-to-be, the micro grid controller is mapped to an operational system in the SOA view. Contrariwise, an operational system, which is also directly integrated the the smart grid itself might be mapped to a micro grid controller. In case, the operational system plays a very central role in the SOA, it might be necessary to understand its smart grid context. Hence, it can be the reason for a refinement. Note that there might be operational systems in the SOA view, which cannot be mapped to any Smart Grid Pattern element, as they are are not concerned with the smart grid at all. The other way round, a micro grid controller might not be mapped at all, as it might not be related to the system-to-be build as SOA. **Source** ☒Thought Experiment ☐ Own Experience ☒ Expert Experience ☐ Documentation **Example(s)** –
Form 24
Source Element Micro Grid Sub Controller **Target Element** Operational System **Relation Type** ☐ Same ☒ Mapped **Refinement** ☐ Source Refined ☒ Target Refined **Reliability** ☐ Unreliable ☒ Reliable **Reasoning** In case a micro grid sub controller is not the central system-to-be but is used as operational system for the system-to-be, the micro grid sub controller is mapped to an operational system in the SOA view. Contrariwise, an operational system, which is also directly integrated the the smart grid itself might be mapped to a micro grid sub controller. In case, the operational system plays a very central role in the SOA, it might be necessary to understand its smart grid context. Hence, it can be the reason for a refinement. Note that there might be operational systems in the SOA view, which cannot be mapped to any Smart Grid Pattern element, as they are are not concerned with the smart grid at all. The other way round, a micro grid sub controller might not be mapped at all, as it might not be related to the system-to-be build as SOA. **Source** ☒Thought Experiment ☐ Own Experience ☒ Expert Experience ☐ Documentation **Example(s)** –
Form 25

Continued on next page

Table B.26 – *continued from previous page*

Relation
Source Element Other Device **Target Element** Operational System **Relation Type** ☐ Same ☒ Mapped **Refinement** ☐ Source Refined ☒ Target Refined **Reliability** ☐ Unreliable ☒ Reliable **Reasoning** In case an other device is not the central system-to-be but is used as operational system for the system-to-be, the other device is mapped to an operational system in the SOA view. Contrariwise, an operational system, which is also directly integrated the the smart grid itself might be mapped to an other device. In case, the operational system plays a very central role in the SOA, it might be necessary to understand its smart grid context. Hence, it can be the reason for a refinement. Note that there might be operational systems in the SOA view, which cannot be mapped to any Smart Grid Pattern element, as they are are not concerned with the smart grid at all. The other way round, an other device might not be mapped at all, as it might not be related to the system-to-be build as SOA. **Source** ☒Thought Experiment ☐ Own Experience ☒ Expert Experience ☐ Documentation **Example(s)** –
Form 26
Source Element Meter **Target Element** Operational System **Relation Type** ☐ Same ☒ Mapped **Refinement** ☐ Source Refined ☒ Target Refined **Reliability** ☐ Unreliable ☒ Reliable **Reasoning** In case a meter is not the central system-to-be but is used as operational system for the system-to-be, the meter is mapped to an operational system in the SOA view. Contrariwise, an operational system, which is also directly integrated the the smart grid itself might be mapped to a meter. In case, the operational system plays a very central role in the SOA, it might be necessary to understand its smart grid context. Hence, it can be the reason for a refinement. Note that there might be operational systems in the SOA view, which cannot be mapped to any Smart Grid Pattern element, as they are are not concerned with the smart grid at all. The other way round, a meter might not be mapped at all, as it might not be related to the system-to-be build as SOA. **Source** ☒Thought Experiment ☐ Own Experience ☒ Expert Experience ☐ Documentation **Example(s)** –
Form 27

Continued on next page

Table B.26 – *continued from previous page*

Relation
Source Element Communication Hub **Target Element** Operational System **Relation Type** ☐ Same ☒ Mapped **Refinement** ☐ Source Refined ☒ Target Refined **Reliability** ☐ Unreliable ☒ Reliable **Reasoning** In case a communication hub is not the central system-to-be but is used as operational system for the system-to-be, the communication hub is mapped to an operational system in the SOA view. Contrariwise, an operational system, which is also directly integrated the the smart grid itself might be mapped to a communication hub. In case, the operational system plays a very central role in the SOA, it might be necessary to understand its smart grid context. Hence, it can be the reason for a refinement. Note that there might be operational systems in the SOA view, which cannot be mapped to any Smart Grid Pattern element, as they are are not concerned with the smart grid at all. The other way round, a communication hub not be mapped at all, as it might not be related to the system-to-be build as SOA. **Source** ☒Thought Experiment ☐ Own Experience ☒ Expert Experience ☐ Documentation **Example(s)** –
Form 28
Source Element Access Device **Target Element** Operational System **Relation Type** ☐ Same ☒ Mapped **Refinement** ☐ Source Refined ☒ Target Refined **Reliability** ☐ Unreliable ☒ Reliable **Reasoning** In case an access device is not the central system-to-be but is used as operational system for the system-to-be, the access device is mapped to an operational system in the SOA view. Contrariwise, an operational system, which is also directly integrated the the smart grid itself might be mapped to an access device. In case, the operational system plays a very central role in the SOA, it might be necessary to understand its smart grid context. Hence, it can be the reason for a refinement. Note that there might be operational systems in the SOA view, which cannot be mapped to any Smart Grid Pattern element, as they are are not concerned with the smart grid at all. The other way round, an access device might not be mapped at all, as it might not be related to the system-to-be build as SOA. **Source** ☒Thought Experiment ☐ Own Experience ☒ Expert Experience ☐ Documentation **Example(s)** –
Form 29

Continued on next page

Table B.26 – *continued from previous page*

Relation
Source Element Service Device **Target Element** Operational System **Relation Type** ☐ Same ☒ Mapped **Refinement** ☐ Source Refined ☒ Target Refined **Reliability** ☐ Unreliable ☒ Reliable **Reasoning** In case a service device is not the central system-to-be but is used as operational system for the system-to-be, the service device is mapped to an operational system in the SOA view. Contrariwise, an operational system, which is also directly integrated the the smart grid itself might be mapped to a service device. In case, the operational system plays a very central role in the SOA, it might be necessary to understand its smart grid context. Hence, it can be the reason for a refinement. Note that there might be operational systems in the SOA view, which cannot be mapped to any Smart Grid Pattern element, as they are are not concerned with the smart grid at all. The other way round, a service device might not be mapped at all, as it might not be related to the system-to-be build as SOA. **Source** ☒Thought Experiment ☐ Own Experience ☒ Expert Experience ☐ Documentation **Example(s)** –
Form 30
Source Element Consumer Device **Target Element** Operational System **Relation Type** ☐ Same ☒ Mapped **Refinement** ☐ Source Refined ☒ Target Refined **Reliability** ☐ Unreliable ☒ Reliable **Reasoning** In case a consumer device is not the central system-to-be but is used as operational system for the system-to-be, the consumer device is mapped to an operational system in the SOA view. Contrariwise, an operational system, which is also directly integrated the the smart grid itself might be mapped to a consumer device. In case, the operational system plays a very central role in the SOA, it might be necessary to understand its smart grid context. Hence, it can be the reason for a refinement. Note that there might be operational systems in the SOA view, which cannot be mapped to any Smart Grid Pattern element, as they are are not concerned with the smart grid at all. The other way round, a consumer device might not be mapped at all, as it might not be related to the system-to-be build as SOA. **Source** ☒Thought Experiment ☐ Own Experience ☒ Expert Experience ☐ Documentation **Example(s)** –

B.2.2. Cloud System Analysis Pattern to SOA Stakeholder Pattern

	Only in Cloud System Analysis Pattern(CSAP)	same	Mapping		Only SOA Layer Stakeholder Pattern (SLSP)
			Element in CSAP	Element in SLSP	
Area					Inner System, Outer System
Machine	Cloud		*SaaS* *IaaS* *PaaS* *Cloud Software Stack* *Software Product*	**Machine**	
Direct Environment		Direct Environment			
Indirect Environment		Indirect Environment			
Layer					Business Domain, Business Processes, Business Services, Infrastructure Services, Component-based Service Realization, Operational System
Process					Process
Relation	has, monitors, builtAndCustomizedBy, buildBy, work for, owns, provides, isComplementedb, isBasedOn, partOf, isA				influences, part of, participates in, provides
Indirect Stakeholder		Domain, Legislator	*Cloud Provider* *Cloud Customer* *End Customer*	Organization	Shareholder, Asset Provider
Direct Stakeholder	Cloud Developer		Cloud Provider	Infrastructure Service Provider *Organization* Process Actor	Component Provider, Operational Systems Provider
			Cloud Customer	Business Service Provider *Organization* Process Actor	
			End Customer	Process Actor *Organization*	
Resource	Resource, Pool				
Active Resource	Hardware, Software, Service		**IaaS**	Infrastructure Service *Machine*	Component
			Cloud Software Stack	Infrastructure Service *Machine*	
			PaaS	Infrastructure Service *Machine*	
			Software Product	Business Service Infrastructure Service CRM ERP Database Packaged Applications Legacy Applications *Machine*	
			SaaS	Business Service *Machine*	
Passive Resource	Data, Location				

Table B.27.: *Cloud System Analyses Pattern to SOA Layer Stakeholder Pattern Relation Investigation Table*

B.2.3. Cloud System Analysis Pattern to Peer to Peer Pattern

	Only in Cloud System Analysis Pattern(CSAP)	same	Mapping		Only Peer to Peer Pattern (PPP)
			Element in CSAP	Element in PPP	
Area					Services, Peer to Peer Protocol
Machine	Cloud				
Direct Environment	Direct Environment				
Indirect Environment	Indirect Environment				
Layer					Application Layer, Service Layer, Feature Management Layer, Overlay Management Layer, Network Layer
Relation	has, monitors, builtAndCustomizedBy, buildBy, work for, owns, provides, isComplementedb, isBasedOn, partOf, isA				uses, constrains
Indirect Stakeholder	Domain, Legislator				
Direct Stakeholder	Cloud Developer, Cloud Provider, Cloud Customer, End Customer				
Resource	Resource, Pool				
Active Resource	Hardware, Software, Service		**IaaS**	Application	Service Management, Service Messaging, Security Management, Reliability and Fault Resilience, Performance and Resource Management, Resource Discovery, Location Lookup, Routing, Network
			Cloud Software Stack	Application	
			PaaS	Application	
			Software Product	Application	
			SaaS	Application	
Passive Resource	Data, Location				Meta Data

Table B.28.: *Cloud System Analysis Pattern to Peer to Peer Pattern Relation Investigation Table*

B.2.4. Peer to Peer Pattern to SOA Stakeholder Pattern

	Only in Peer to Peer Pattern (PPP)	same	Mapping		Only SOA Layer Stakeholder Pattern (SLSP)
			Element in PPP	Element in SLSP	
Area	Services, Peer to Peer Protocol				Inner System, Outer System
Machine			*Application*	**Machine**	
Direct Environment					Direct Environment
Indirect Environment					Indirect Environment
Layer	Application Layer, Service Layer, Feature Management Layer, Overlay Management Layer, Network Layer				Business Domain, Business Processes, Business Services, Infrastructure Services, Component-based Service Realization, Operational System
Process					Process
Relation	uses, constrains				influences, part of, participates in, provides
Indirect Stakeholder					Domain, Legislator, Shareholder, Asset Provider, Organization
Direct Stakeholder					Component Provider, Operational Systems Provider, Infrastructure Service Provider, Business Service Provider, Process Actor
Active Resource	Service Management, Service Messaging, Security Management, Reliability and Fault Resilience, Performance and Resource Management, Resource Discovery, Location Lookup, Routing, Network		Application	**Infrastructure Service**	Business Service, CRM, ERP, Database, Packaged Applications, Legacy Applications, Component
				Machine	
Passive Resource	Meta Data				

Table B.29.: *Peer to Peer Pattern to SOA Stakeholder Pattern Relation Investigation Table*

B.2.5. SOA Stakeholder Pattern To SOA Layer Pattern

	Only in SOA Layer Pattern (SLP)	same	Mapping		Only SOA Layer Stakeholder Pattern (SLSP)
			SLP	Element in SLSP	
Area					Inner System, Outer System
Machine			*Whole Pattern*	**Machine**	
Direct Environment					Direct Environment
Indirect Environment					Indirect Environment
Layer		Business Domain, Business Processes, Business Services, Infrastructure Services, Component-based Service Realization, Operational System			
Process		Process			
Relation	performed by, relies on, exposes, business relation	participates in			influences, part of, provides
Indirect Stakeholder		Organization			Domain, Legislator,Shareholder, Asset Provider
Direct Stakeholder					Component Provider, Operational Systems Provider, Infrastructure Service Provider, Process Actor, Business Service Provider
Active Resource		Component, Business Service, CRM, ERP, Database, Packaged Applications, Legacy Applications, Infrastructure Service			

Table B.30.: *SOA Stakeholder Pattern To SOA Layer Pattern Relation Investigation Table*

B.2.6. Law Pattern to Law Identification Pattern

	Only in Law Pattern (LP)	same	Mapping		Only Law Identification Pattern (LIP)
			Element in LP	Element in LIP	
Area		Classification			
Direct Environment			Law Structure	Core Structure	
Indirect Environment			Context	Context	
Process					Related Process(es)
Activities		Activities, Activity Classifier			
Relation	isA	Avoid, Accomplish, Influence, Entitled To			Classified As
Stakeholder		Person Classifier			
Indirect Stakeholder		Domain, Legislator			
Direct Stakeholder			Addressee	Active Stakeholder	
			Target Person	Passive Stakeholder	
Resource		Subject Classifier	Target Subject	Asset	
Passive Resource	Regulations				

Table B.31.: *Law Pattern to Law Identification Pattern Relation Investigation Table*

B.2.7. SOA Stakeholder Pattern To Law Identification Pattern

	Only SOA Layer Stakeholder Pattern (SLSP)	same	Mapping		Only Law Identification Pattern (LIP)
			Element in SLSP	Element in LIP	
Area	Inner System, Outer System				Classification
Machine			Machine	*Asset*	
Direct Environment			Direct Environment	Core Structure	
Indirect Environment			Indirect Environment	Context	
Layer		Business Domain, Business Processes, Business Services, Infrastructure Services, Component-based Service Realization, Operational System			
Process			Process	Related Process(es)	
Activities					Activities, Activity Classifier
Relation	performed by, relies on, exposes, business relation, participates in				Avoid, Accomplish, Influence, Entitled To, Classified As
Stakeholder					Person Classifier
Indirect Stakeholder		Domain, Legislator	Asset Provider	*Passive Stakeholder*	
			Shareholder	*Passive Stakeholder*	
			Organization	*Passive Stakeholder*	
				Active Stakeholder	
Direct Stakeholder			Component Provider	Active Stakeholder	
			Operational Systems Provider		
			Infrastructure Service Provider		
			Process Actor		
			Business Service Provider		
			Organization		
			Component Provider	Passive Stakeholder	
			Operational Systems Provider		
			Infrastructure Service Provider		
			Process Actor		
			Business Service Provider		
			Asset Provider		
			Shareholder		
			Organization		
Resource					Subject Classifier, Asset
Active Resource			Component	*Asset*	
			Business Service		
			CRM		
			ERP		
			Database		
			Packaged Applications		
			Legacy Applications		
			Infrastructure Service		

Table B.32.: *SOA Stakeholder Pattern To Law Identification Pattern Relation Investigation Table*

B.2.8. Cloud System Analysis Pattern to Law Identification Pattern

	Only in Cloud System Analysis Pattern(CSAP)	same	Mapping		Only Law Identification Pattern (LIP)
			Element in CSAP	Element in LIP	
Area					Classification
Machine			Cloud	*Asset*	
Direct Environment			Direct Environment	Core Structure	
Indirect Environment			Indirect Environment	Context	
Process					Related Process(es)
Activities					Activities, Activity Classifier
Relation	has, monitors, builtAndCustomizedBy, buildBy, work for, owns, provides, isComplementedb, isBasedOn, partOf, isA				Avoid, Accomplish, Influence, Entitled To, Classified As
Stakeholder					Person Classifier
Indirect Stakeholder		Domain, Legislator			
Direct Stakeholder			Cloud Provider	Active Stakeholder	
			Cloud Customer		
			End Customer		
			Cloud Developer		
			Cloud Provider	Passive Stakeholder	
			Cloud Customer		
			End Customer		
			Cloud Developer		
Resource			Resource	Asset	Subject Classifier
			Pool		
Active Resource			IaaS	*Asset*	
			Cloud Software Stack		
			PaaS		
			Software Product		
			SaaS		
			Hardware		
			Software		
			Service		
Passive Resource	Location		Data	*Asset*	

Table B.33.: *Cloud System Analysis Pattern to Law Identification Pattern Relation Investigation Table*

Appendix C

Goal Elicitation

The decision was made using the questions and recommended notations for an answer as proposed by Horkoff and Yu (2011 [188]). Table C.1 shows the result.

Category	Question	Answer	Recommended Notations
Domain Understanding	QU1. Does the domain contain a high degree of social interaction, have many stakeholders with differing goals, or involve many interacting systems?	**Yes.** Our method shall be able to cover such scenarios in general, and our running example is an instance of such a scenario.	**i***, GRL, Tropos
	QU2. Do you need to understand details of the system at this point? Do you have access to detailed information such as cost, probabilities, and conditions? Can you express necessary or desired domain properties?	**No.** We just need the goals. The details are elicited and modeled later on.	no recommendation
Communication	QC1. Do you need to communicate with stakeholders? Validate requirements in the model? Justify recommendations?	**Yes.** In general, it is the aim of the method to elicit the goals in cooperation with actual stakeholders if possible.	**i***, KAOS, NFR, Tropos, GRL
Model Improvement	QM1. Are you confident in the accuracy, structure, and completeness of domain knowledge and models?	**No.** We already elicited the context, but may have missed detailed domain knowledge up to this point.	**i***, SNET, NFR, Tropos
	QM2. Would you like to verify critical properties over the model?	**No.** At this point we are not looking for specific critical properties.	no recommendation
Scoping	QS1. Do you need to determine system scope?	**No.** The scope is already known from the context elicitation.	no recommendation
Requirements Elicitation	QE1. Do you need to find more high-level requirements? Are you looking for ways to prompt further elicitation?	**Yes.** We are not looking for detailed requirements at this point in the process.	**i***, SNET, NFR, Tropos
	QE2. Do you need to find detailed system requirements?	**No.** We are not looking for detailed requirements at this point in the process.	no recommendation
	QE3. Do you need to consider non-functional requirements difficult to quantify?	**No.** We do not need a quantification at this point.	no recommendation
	QE4. Do you need to capture domain assumptions?	**Yes.** Might be handy to already capture assumptions done at this point.	**i***
Requirements Improvement	QR1. Are you working with a system where safety/security/ privacy/risks or other specific properties are critical considerations?	**No.** At this point we are not looking for specific critical properties.	no recommendation
	QR2. Do you need to find errors and inconsistencies in requirements?	Not relevant	no recommendation
Design	QD1. Are you aware of a sufficient number of high-level design alternatives?	Not relevant	no recommendation
	QD2. Are you aware of a sufficient number of detailed design alternatives?	Not relevant	no recommendation
	QD3. Do you need to evaluate and choose between high-level design alternatives?	Not relevant	no recommendation
	QD4. Do you need to evaluate and choose between detailed design alternatives?	Not relevant	no recommendation
	QD5. Do you need to find acceptable processes?	Not relevant	no recommendation
	QD6. Do you need to test run-time operation before implementation?	Not relevant	no recommendation

Table C.1.: *Decision for i**

Appendix D

Problem Context and Functional Requirements Elicitation

Appendix D.1 shows the transformation tables for transforming EPCs to context diagrams. Appendix D.2[1] shows the transformation tables for transforming FADs to context diagrams.

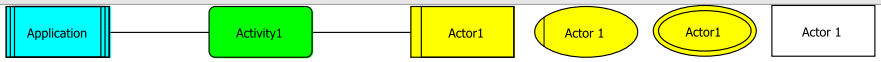
D.1. Context Diagram(s)

[1] Page 475

Case 1: Application, Function, Position | Group | Organizational Unit | External Person

Input

Part of the EPC

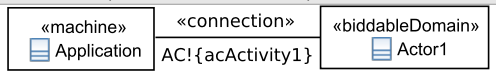


Questions (for semi-automated transformation)

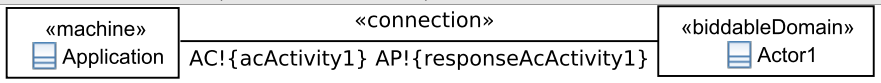
(1)	Can the *Actor1* trigger the *Activity1*?	☐
(1.1)	Gets the *Actor1* a response by the *Application* when he/she triggers the *Activity1*?	☐
(2)	Can the *Application* trigger the *Activity1*?	☐
(2.1)	Gets the *Application* a response by the *Actor1* when it triggers the *Activity1*?	☐

Output

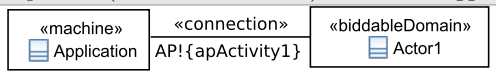
Option 1 (**1**☑ **1.1**☒ **2**☒ **2.1**☒): External Trigger



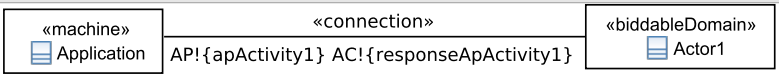
Option 2 (**1**☑ **1.1**☑ **2**☒ **2.1**☒): External Trigger with Response



Option 3 (**1**☒ **1.1**☒ **2**☑ **2.1**☒): Internal Trigger



Option 4 (**1**☒ **1.1**☒ **2**☑ **2.1**☑): Internal Trigger with Response



Option 5 (**1**☑ / ☐ **1.1**☑ / ☐ **2**☑ / ☐ **2.1**☑ / ☐): Multi-Trigger with Response

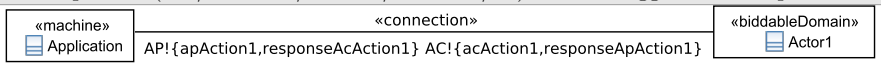
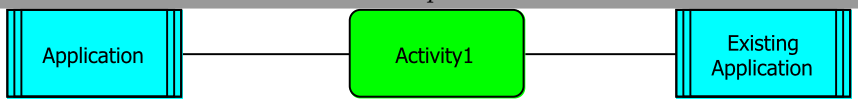


Table D.1.: *Create Phenomena: Case 1*

Case 2: Application, Function, Existing Application

Input

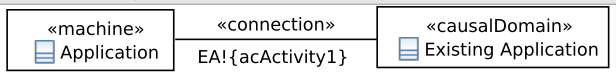


Questions (for semi-automated transformation)

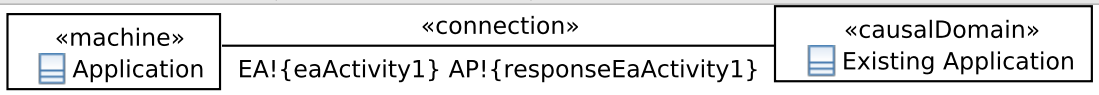
(1)	Can the *Existing Application* trigger the *Activity1*?	☐
(1.1)	Gets the *Existing Application* a response by the *Application* when it triggers the *Activity1*?	☐
(2)	Can the *Application* trigger the *Activity1*?	☐
(2.1)	Gets the *Application* a response by the *Existing Application* when it triggers the *Activity1*?	☐

Output

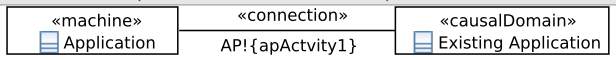
Option 1 (**1**☑ **1.1**☐ **2**☐ **2.1**☐): External Trigger



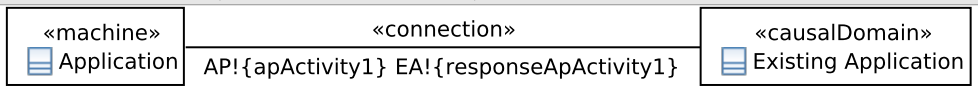
Option 2 (**1**☑ **1.1**☑ **2**☐ **2.1**☐): External Trigger with Response



Option 3 (**1**☐ **1.1**☐ **2**☑ **2.1**☐): Internal Trigger



Option 4 (**1**☐ **1.1**☐ **2**☑ **2.1**☑): Internal Trigger with Response



Option 5 **1**☑/☐ **1.1**☑/☐ **2**☑/☐ **2.1**☑/☐): Multi-Trigger with Response

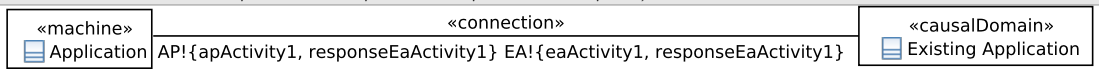
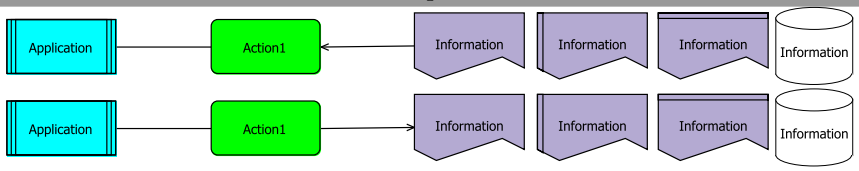


Table D.2.: *Create Phenomena: Case 2*

Case 3: Application, Function, Document | List | Log | File

Input



Output

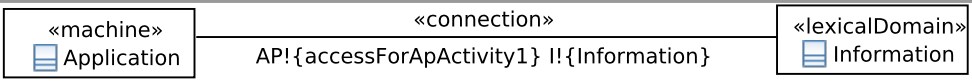
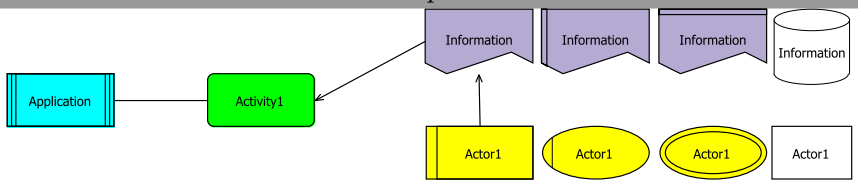


Table D.3.: *Create Phenomena: Case 3*

Case 4: Application, Function, Document | List | Log | File, Position | Group | Organizational Unit | External Person

Input

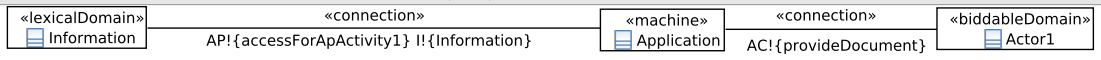


Questions (for semi-automated transformation)

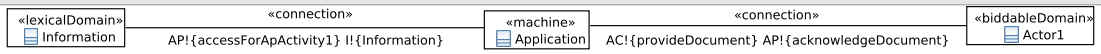
(1) Does the *Actor1* get an acknowledgment for the provided *Information* by the *Application* ? □

Output

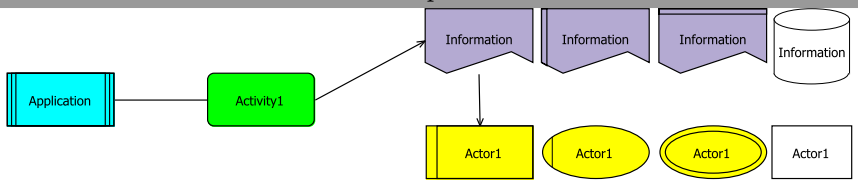
Option 1 (1☑): Provided Information



Option 2 (1☑ / ☐): Provided Information with Acknowledgment



Input



Questions (for semi-automated transformation)

(1) Does the *Application* get an acknowledgment for the provided *Information* by the *Actor1* ? □

Output

Option 3 (1☑): Provide Information

Option 4 (1☑ / ☐): Provide Information with Acknowledgment

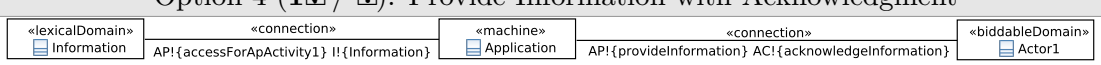
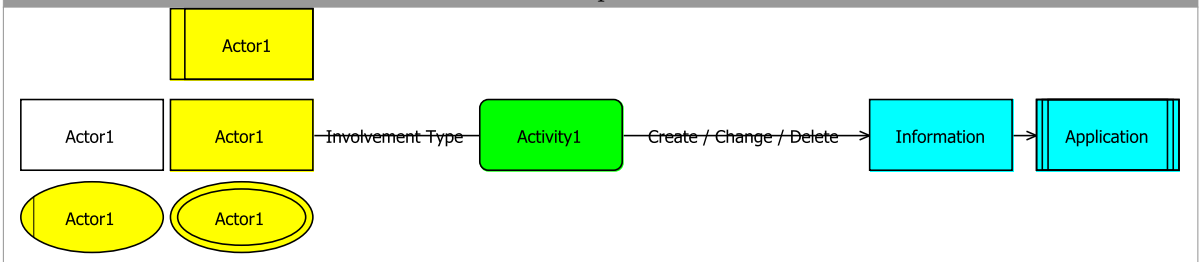


Table D.4.: *Create Phenomena: Case 4*

D.2. Problem Diagram(s)

Case 1: Application and operations *create, update, delete*

Input

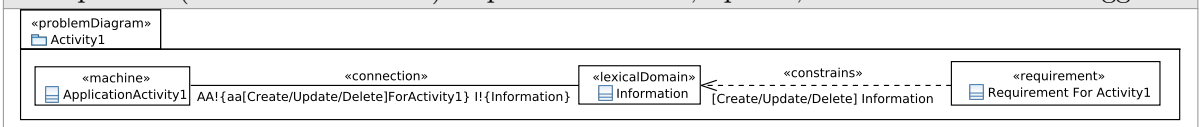


Questions (for semi-automated transformation)

(1)	Can the *Actor1* trigger the operation *create* | *update* | *delete* on *Information* for *Activity1*?	☐
(1.1)	Gets the *Actor1* a response by the *Application* when he/she triggers the operation *create* | *update* | *delete* on *Information* for the *Activity1*?	☐
(2)	Can the *Application* trigger the operation *create* | *update* | *delete* on *Information* for *Activity1*?	☐
(2.1)	Gets the *Actor1* a notification by the *Application* when it triggers the operation *create* | *update* | *delete* on *Information* for *Activity1*?	☐

Output

Option 1 (**1**🗲 **1.1**🗲 **2**☑ **2.1**🗲): Operations create, update, delete with internal trigger



Option 2 (**1**🗲 **1.1**🗲 **2**☑ **2.1**☑): Operations create, update, delete with internal trigger and notification

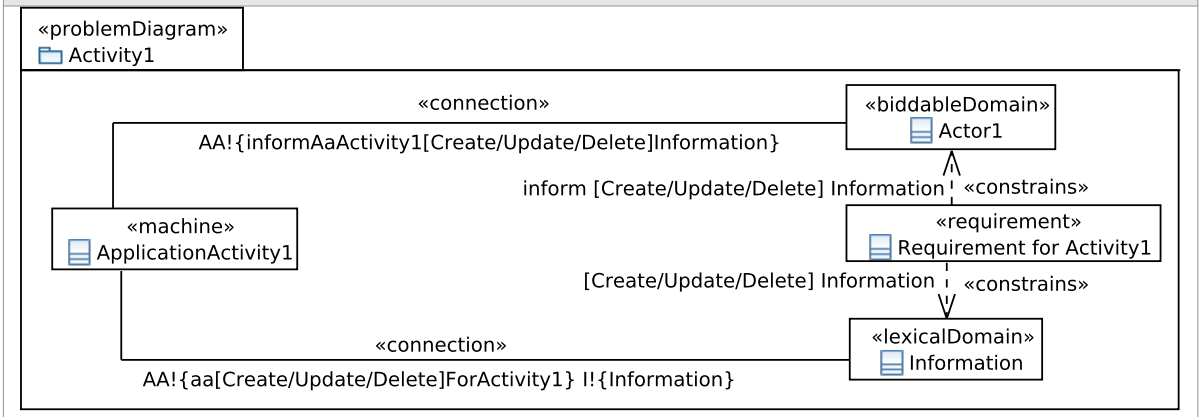
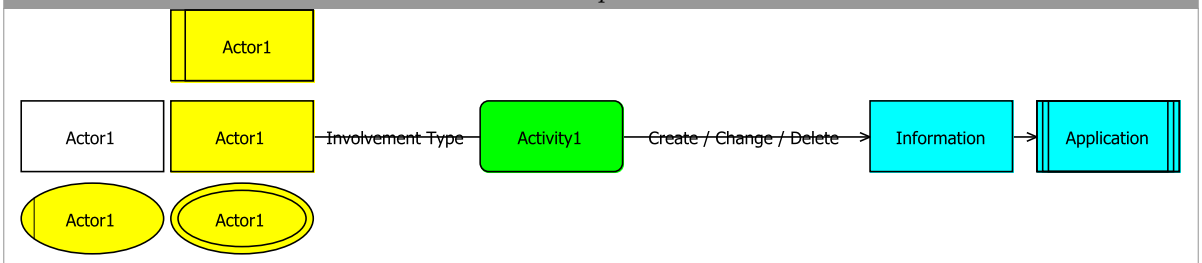


Table D.5.: *Create Phenomena: Case 1 (1/2)*

Case 1: Application and operations *create, update, delete*

Input

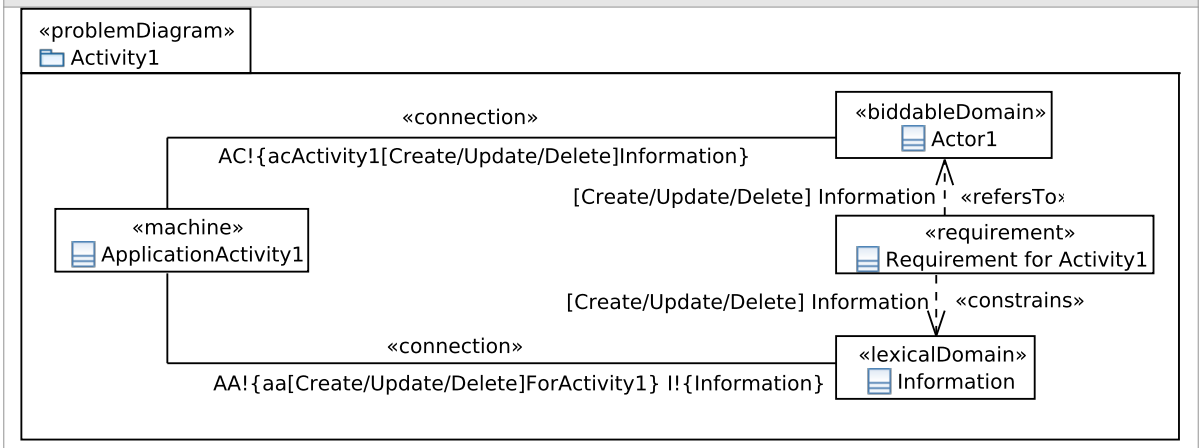


Questions (for semi-automated transformation)

(1)	Can the *Actor1* trigger the operation *create* | *update* | *delete* on *Information* for *Activity1*?	☐
(1.1)	Gets the *Actor1* a response by the *Application* when he/she triggers the operation *create* | *update* | *delete* on *Information* for the *Activity1*?	☐
(2)	Can the *Application* trigger the operation *create* | *update* | *delete* on *Information* for *Activity1*?	☐
(2.1)	Gets the *Actor1* a notification by the *Application* when it triggers the operation *create* | *update* | *delete* on *Information* for *Activity1*?	☐

Output

Option 3 (**1**☑ **1.1**☒ **2**☒ **2.1**☒): Operations create, update, delete with external trigger by actor



Option 4 (**1**☑ **1.1**☑ **2**☒ **2.1**☒): Operations create, update, delete with external trigger by actor and response

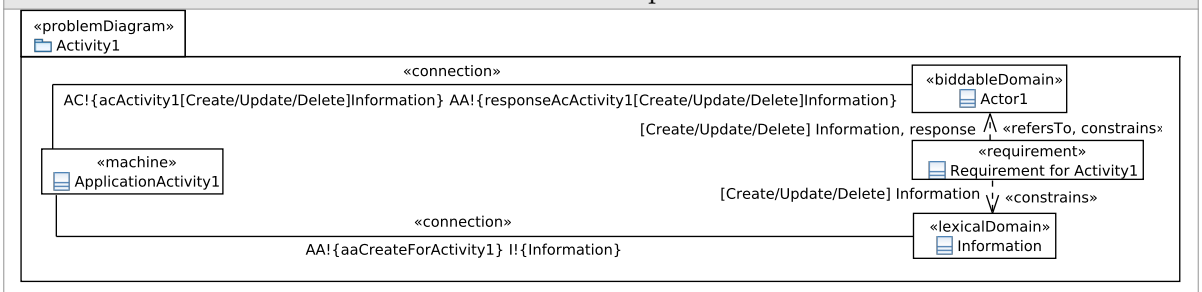
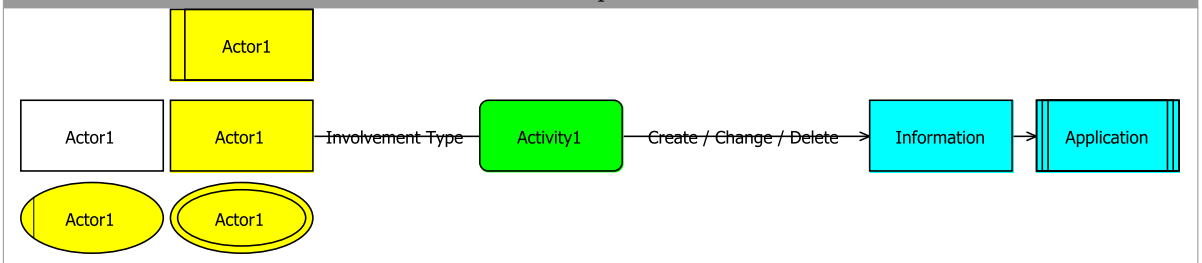


Table D.6.: *Create Phenomena: Case 1 (2/3)*

Case 1: Application and operations *create, update, delete*

Input



Questions (for semi-automated transformation)

(1)	Can the *Actor1* trigger the operation *create* | *update* | *delete* on *Information* for *Activity1*?	☐
(1.1)	Gets the *Actor1* a response by the *Application* when he/she triggers the operation *create* | *update* | *delete* on *Information* for the *Activity1*?	☐
(2)	Can the *Application* trigger the operation *create* | *update* | *delete* on *Information* for *Activity1*?	☐
(2.1)	Gets the *Actor1* a notification by the *Application* when it triggers the operation *create* | *update* | *delete* on *Information* for *Activity1*?	☐

Option 5 (**1**☑/ ☐ **1.1**☑/ ☐ **2**☑/ ☐ **2.1**☑/ ☐): Operations create, update, delete with multiple triggers

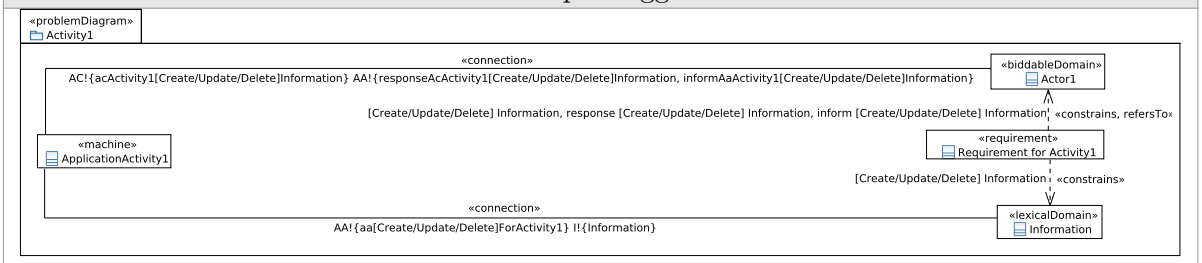
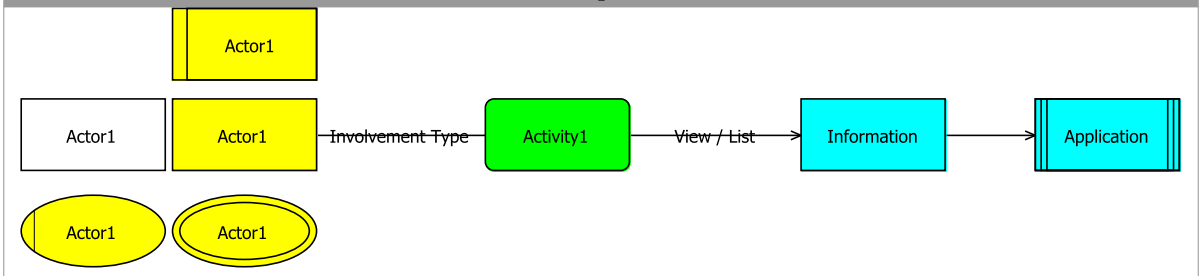


Table D.7.: *Create Phenomena: Case 1 (3/3)*

Case 2: Application and operations *view, list*

Input



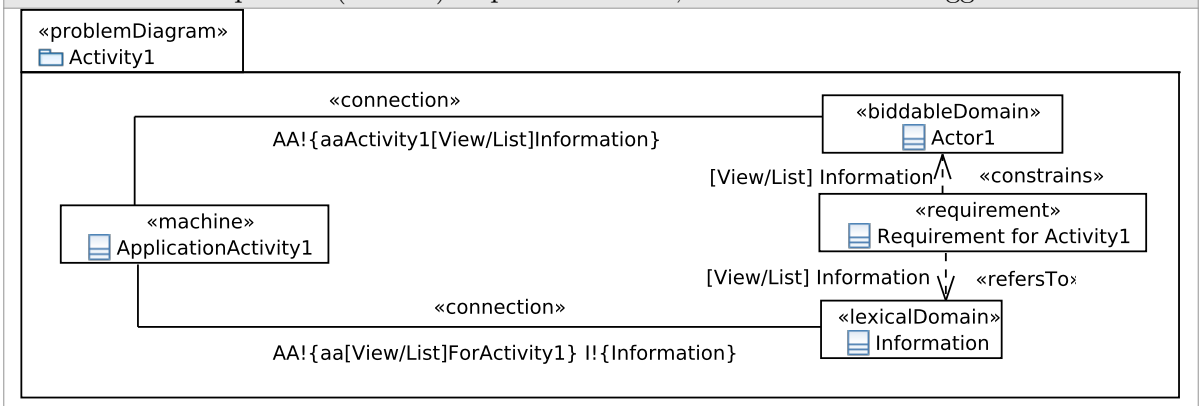
Questions (for semi-automated transformation)

(1) Can the *Actor1* trigger the operation *view* | *list* on *Information* for *Activity1*? ☐

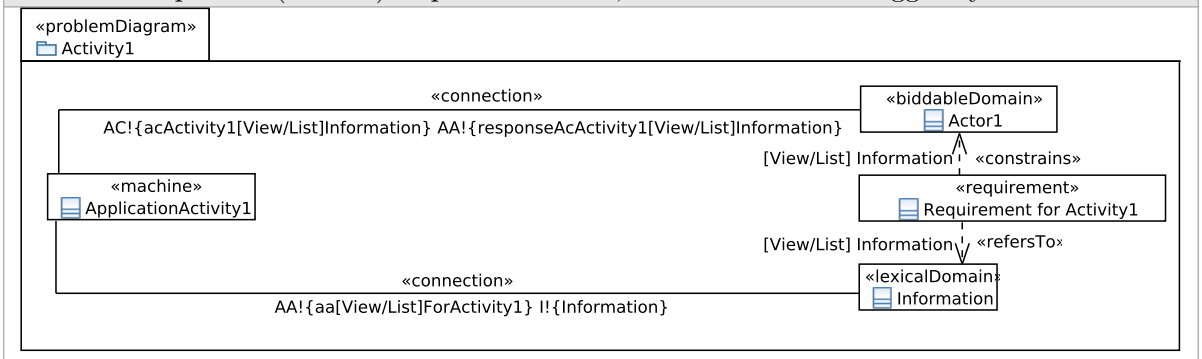
(2) Can the *Application* trigger *view* | *list* on *Information* for *Activity1*? ☐

Output

Option 1 (**1**☑ **2**☑): Operations view, list with internal trigger



Option 2 (**1**☑ **2**☑): Operations view, list with external trigger by actor



Option 3 (**1**☑/☐ **2**☑/☐): Operations list, view with multiple triggers

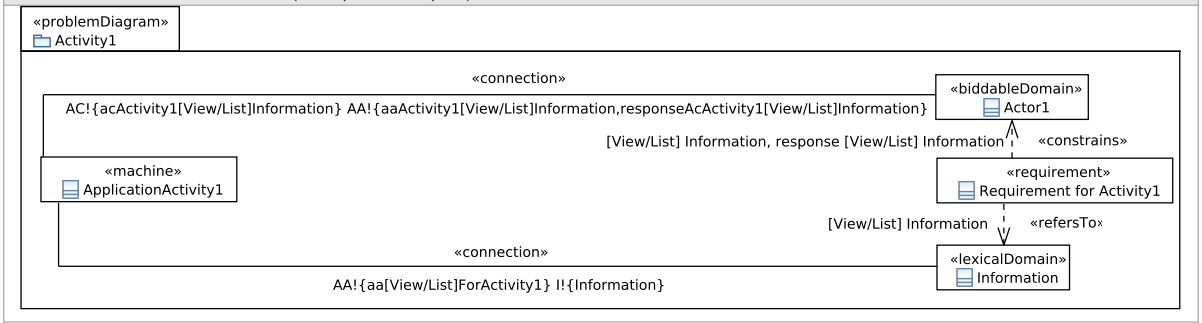
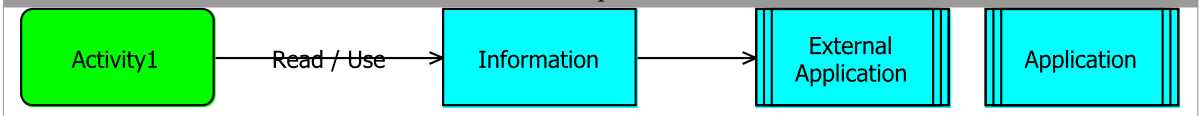


Table D.8.: *Create Phenomena: Case 2*

Case 3: (External) application and operations *read, use*

Input



Output

Option 1 (Default Case): Operations use, read with internal trigger

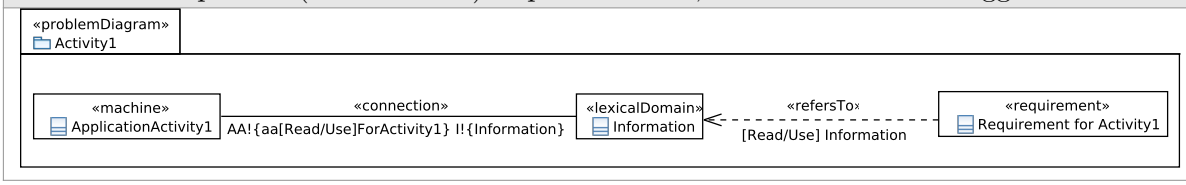
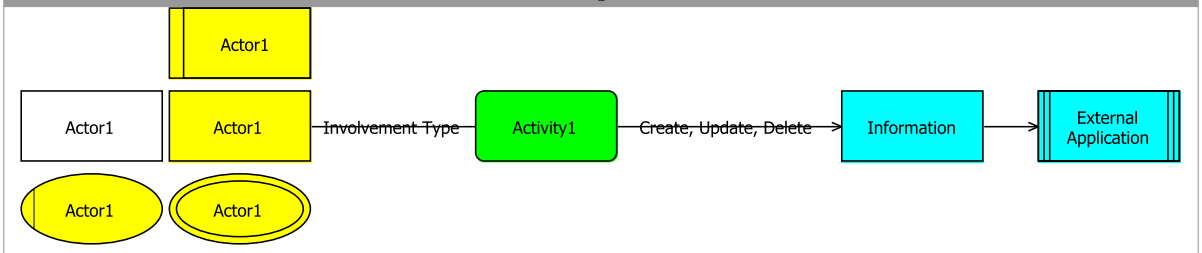


Table D.9.: *Create Phenomena: Case 3*

Case 4: External Application and operation *create, update, delete*

Input

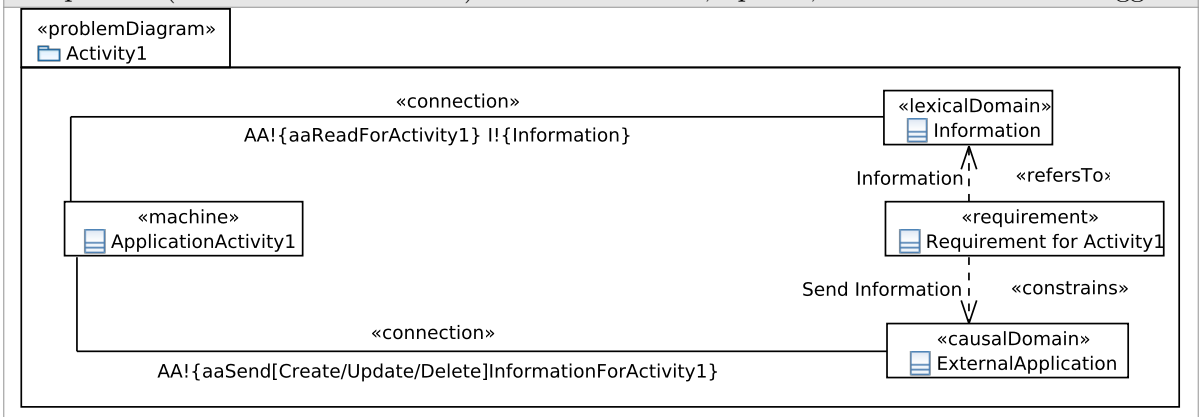


Questions (for semi-automated transformation)

(1)	Can the *Actor1* trigger the operation *create* | *update* | *delete* on *Information* for *Activity1*?	☐
(1.1)	Gets the *Actor1* a response by the *Application* when he/she triggers the operation *create* | *update* | *delete* on *Information* for the *Activity1*?	☐
(2)	Can the *Application* trigger the operation *create* | *update* | *delete* on *Information* for *Activity1*?	☐
(2.1)	Gets the *Actor1* a notification by the *Application* when it triggers the operation *create* | *update* | *delete* on *Information* for *Activity1*?	☐
(3)	Does the *External Application* acknowledge the operation *create* | *update* | *delete* on *Information* for *Activity1*?	☐

Output

Option 1 (**1☓ 1.1☓ 2☑ 2.1☓ 3☓**) : Function create, update, delete with internal trigger



Option 2 (**1☓ 1.1☓ 2☑ 2.1☓ 3☑**): Function create, update, delete with internal trigger and response

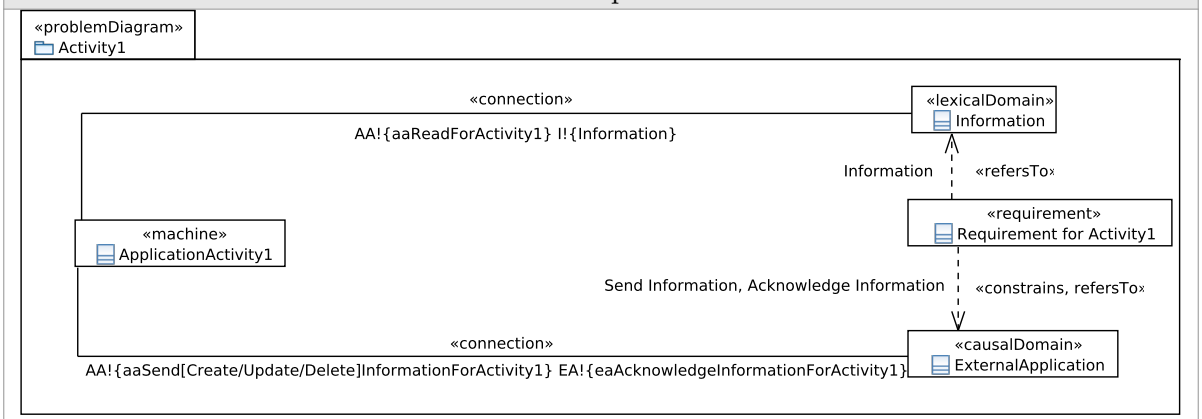
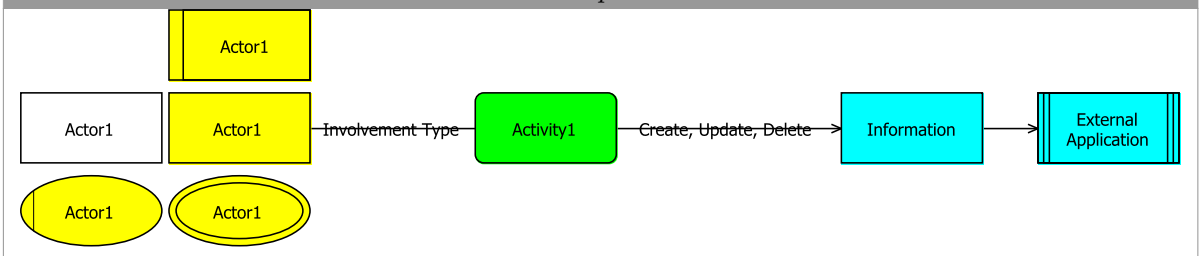


Table D.10.: *Create Phenomena: Case 4 (1/4)*

Case 4: External Application and operation *create, update, delete*

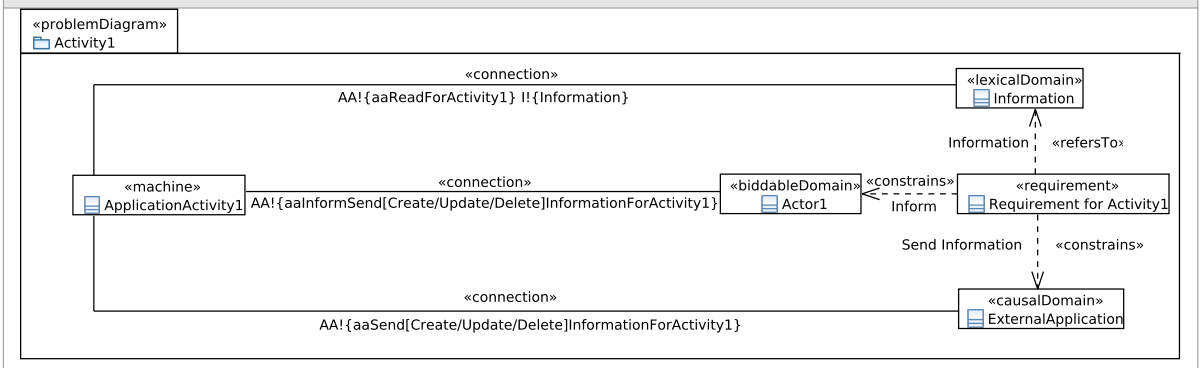
Input



Questions (for semi-automated transformation)

(1)	Can the *Actor1* trigger the operation *create* | *update* | *delete* on *Information* for *Activity1*?	☐
(1.1)	Gets the *Actor1* a response by the *Application* when he/she triggers the operation *create* | *update* | *delete* on *Information* for the *Activity1*?	☐
(2)	Can the *Application* trigger the operation *create* | *update* | *delete* on *Information* for *Activity1*?	☐
(2.1)	Gets the *Actor1* a notification by the *Application* when it triggers the operation *create* | *update* | *delete* on *Information* for *Activity1*?	☐
(3)	Does the *External Application* acknowledge the operation *create* | *update* | *delete* on *Information* for *Activity1*?	☐

Option 3 (**1**☓ **1.1**☓ **2**☑ **2.1**☑ **3**☓): Function create, update, delete with internal trigger, and inform



Option 4 (**1**☓ **1.1**☓ **2**☑ **2.1**☑ **3**☑): Function create, update, delete with internal trigger, response and inform

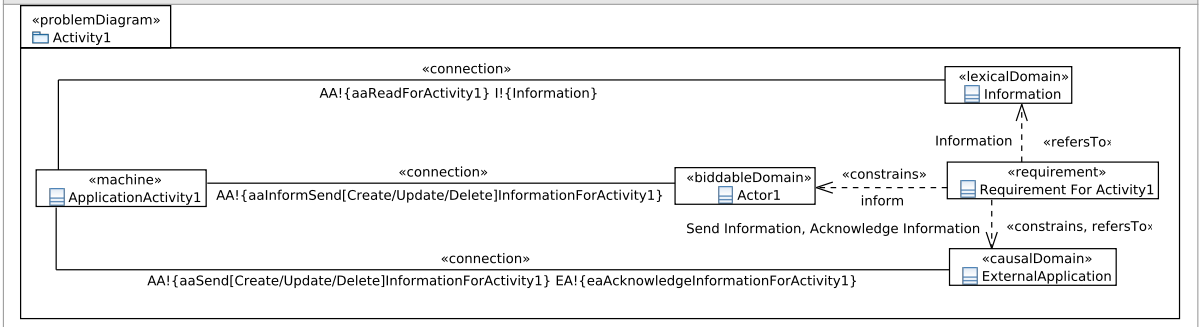
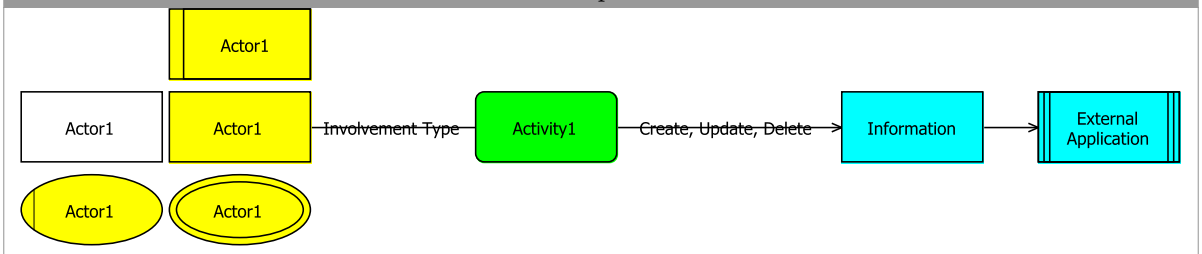


Table D.11.: *Create Phenomena: Case 4 (2/4)*

Case 4: External Application and operations *create, update, delete*

Input

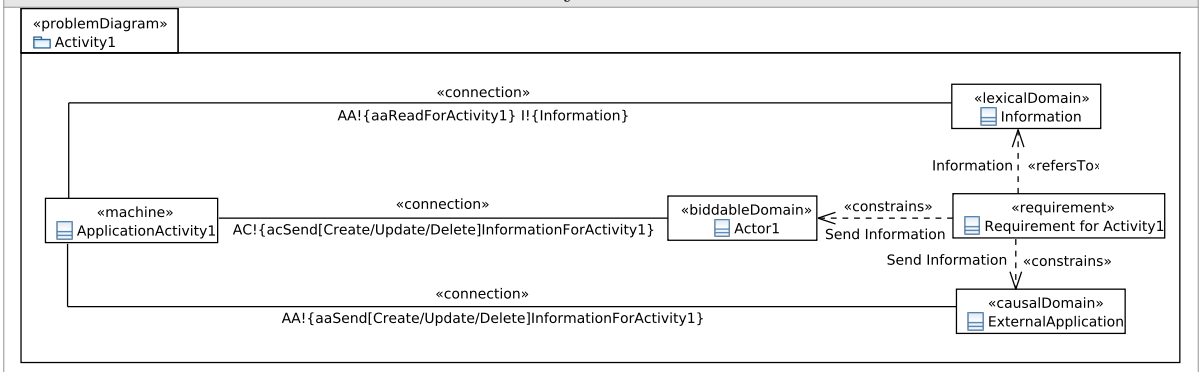


Questions (for semi-automated transformation)

(1)	Can the *Actor1* trigger the operation *create* | *update* | *delete* on *Information* for *Activity1*?	☐
(1.1)	Gets the *Actor1* a response by the *Application* when he/she triggers the operation *create* | *update* | *delete* on *Information* for the *Activity1*?	☐
(2)	Can the *Application* trigger the operation *create* | *update* | *delete* on *Information* for *Activity1*?	☐
(2.1)	Gets the *Actor1* a notification by the *Application* when it triggers the operation *create* | *update* | *delete* on *Information* for *Activity1*?	☐
(3)	Does the *External Application* acknowledge the operation *create* | *update* | *delete* on *Information* for *Activity1*?	☐

Output

Option 5 (**1☑ 1.1☐ 2☐ 2.1☐ 3☐**): Operations create, update, delete with external trigger by actor



Option 6 (**1☑ 1.1☐ 2☐ 2.1☐ 3☑**): Operations create, update, delete with external trigger by actor and external application response

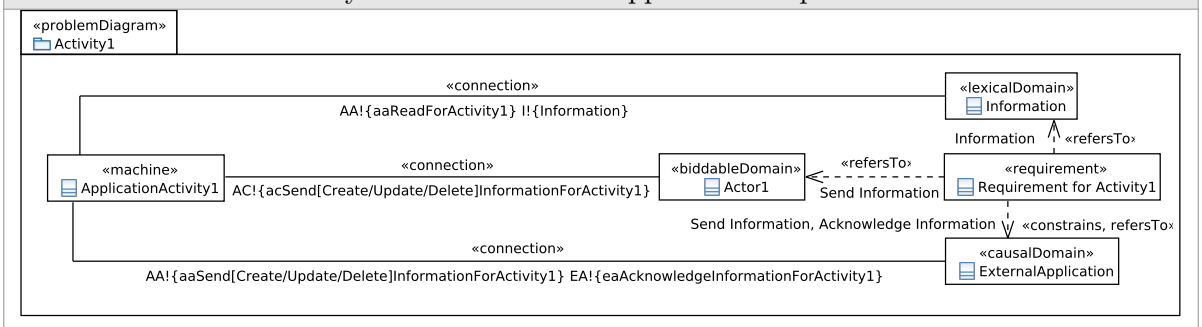
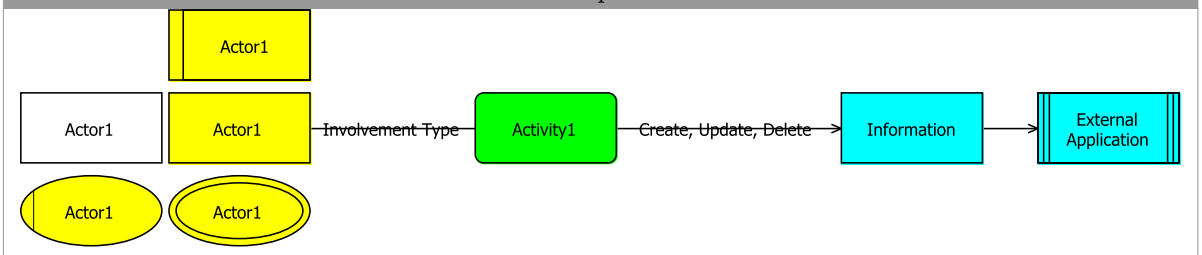


Table D.12.: *Create Phenomena: Case 4 (3/4)*

Case 4: External Application and operations *create, update, delete*

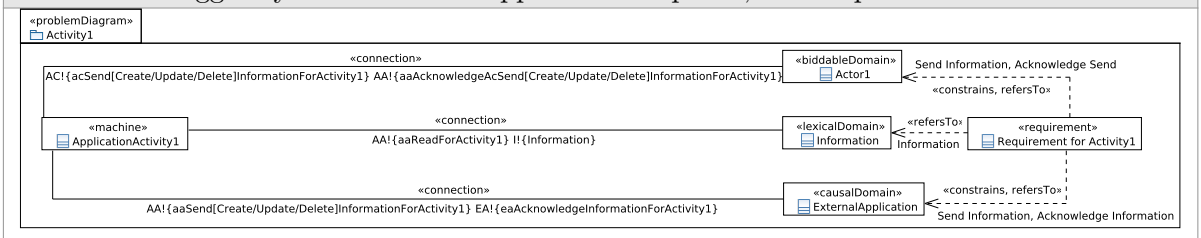
Input



Questions (for semi-automated transformation)

(1)	Can the *Actor1* trigger the operation *create* | *update* | *delete* on *Information* for *Activity1*?	☐
(1.1)	Gets the *Actor1* a response by the *Application* when he/she triggers the operation *create* | *update* | *delete* on *Information* for the *Activity1*?	☐
(2)	Can the *Application* trigger the operation *create* | *update* | *delete* on *Information* for *Activity1*?	☐
(2.1)	Gets the *Actor1* a notification by the *Application* when it triggers the operation *create* | *update* | *delete* on *Information* for *Activity1*?	☐
(3)	Does the *External Application* acknowledge the operation *create* | *update* | *delete* on *Information* for *Activity1*?	☐

Option 7 (**1☑ 1.1☑ 2☒ 2.1☒ 3☑**): Operations create, update, delete with external trigger by actor external application response, and response to actor



Option 8 (**1☑/☐ 1.1☑/☐ 2☑/☐ 2.1☑/☐ 3☑/☐**): Operations create, update, delete with multiple triggers

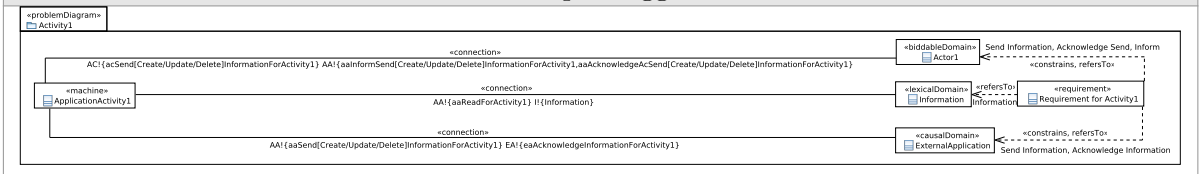
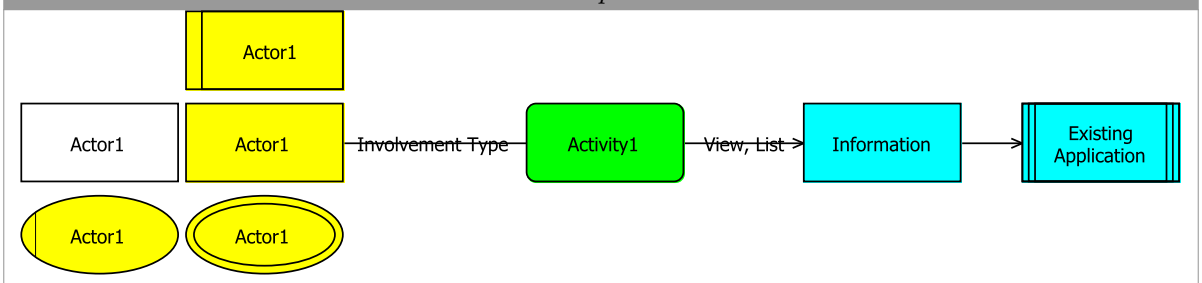


Table D.13.: *Create Phenomena: Case 4 (4/4)*

Case 5: External Application and operations *view, list*

Input



Questions (for semi-automated transformation)

(1)	Can the *Actor1* trigger the operation *view* | *list* on *Information*, which is retrieved from the external application, for *Activity1*?	☐
(1.1)	Gets the *Actor1* a response by the *Application* when he/she triggers the operation *view* | *list* on *Information* for the *Activity1*?	☐
(2)	Can the *Application* trigger the operation *view* | *list* on *Information*, which is retrieved from the external application, for *Activity1*?	☐
(2.1)	Gets the *Actor1* a notification by the *Application* when it triggers the operation *view* | *list* on *Information*, which is retrieved from the external application, for *Activity1*?	☐

Option 1 (**1**☑ **1.1**🔄 **2**🔄 **2.1**🔄): Operations view, list with external trigger by actor

Option 2 (**1**☑ **1.1**☑ **2**🔄 **2.1**🔄): Operations view, list with external trigger by actor and response to actor

Option 3 (**1**🔄 **1.1**🔄 **2**☑ **2.1**🔄): Operations view, list with internal trigger

Option 4 (**1**🔄 **1.1**🔄 **2**☑ **2.1**☑): Operations view, list with internal trigger and inform of actor

Option 5 (**1**☑/❓ **1.1**☑/❓ **2**☑/❓ **2.1**☑/❓): Operations create, update, delete with multiple triggers

Table D.14.: *Create Phenomena: Case 5*

Appendix E

Compliance Requirements Engineering

This appendix contains additional material regarding the method for legal compliance requirements engineering. Appendix E.1 contains a full subsumption example. The full example dictates of justice used in this thesis are shown in Appendix E.2[1].

E.1. Subsumption Method: Full Example

Table E.1 shows the full subsumption example, which was already shown partially in Section 14.4.2[2].

[1] Page 487
[2] Page 253

Hypothesis	The company might have to ensure that specific rights and privacy needs of the participants, who are entitled to the registration data, are preserved.			
Major Premise	**(BDSG Section 1)** The purpose of this Act is to protect the individual against his/her right to privacy being impaired through the handling of his/her personal data. … This Act shall apply to the collection, processing and use of personal data By … private bodies in so far as they process or use data by means of data processing systems or collect data for such systems …			
Subsumption	**Hypothesis 1**	The participants could be in danger of loosing privacy when the registration data is handled improperly.		
	Major Premise 1	**(BDSG Section 1)** The purpose of this Act is to protect the individual against his/her right to privacy being impaired through the handling of his/her personal data.		
	Subsumption 1	**Hypothesis 1.1**	A participant could be an individual.	
		Major Premise 1.1	**(Black's Law Dictionary)** A single person as distinguished from a group or class, and also, a private or natural person as distinguished from a public or private body	
		Subsumption 1.1	A participant is a single, natural person.	
		Conclusion 1.1	*A participant is an individual.*	
		Hypothesis 1.2	Registration data, including full name and address, could be personal data	
		Major Premise 1.2	**(BDSG Section 3)** "Personal data" means any information concerning the personal or material circumstances of an identified or identifiable individual.	
		Subsumption 1.2	**Hypothesis 1.2.1**	The full name might identify an participant or might make him/her identifiable.
			Major Premise 1.2.1	**(Black's Law Dictionary)** *Identification:* Proof of identity; the proving that a person, subject, or article before the court is the very same that he or it is alleged, charged, or reputed to be; True identity is collected from a multitude of signs.
			Subsumption 1.2.1	The full name might not reveal the true identity in all cases. But it is one of the signs which can be used to derive the true identity.
			Conclusion 1.2.1	*The full name identifies an participant or makes him/her identifiable.*
			Hypothesis 1.2.2	The address might describe personal circumstances.
			Major Premise 1.2.2	**(Black's Law Dictionary)** *Personal Circumstances:* Information relating to the private aspects of a person's life.
			Subsumption 1.2.2	The information where somebody is living reveals private aspects of this person's life.
			Conclusion 1.2.2	*The address describes personal circumstances.*
			Registration data includes information about personal circumstances. And it identifies the related person or makes the person identifiable.	
		Conclusion 1.2	*Registration data, including full name and address, is personal data*	
		Hypothesis 1.3	Someone might be able to impair the privacy of the participant using the registration data.	
		Major Premise 1.3	**(Black's Law Dictionary)** *Privacy:* The right that determines the nonintervention of secret surveillance and the protection of an individual's information. It is split into 4 categories (1) Physical: An imposition whereby another individual is restricted from experiencing an individual or a situation. (2) Decisional: The imposition of a restriction that is exclusive to an entity. (3) Informational: The prevention of searching for unknown information and (4) Dispositional: The prevention of attempts made to get to know the state of mind of an individual.	
		Subsumtion 1.3	As personal data is involved (See above), somebody could search and find unknown information.	
		Conclusion 1.3	*Someone can impair the privacy of the participant using the registration data.*	
		When the registration data leaks to unauthorized persons, the unauthorized person can tamper with the privacy of the participant.		
	Conclusion 1	*The participants is in danger of loosing privacy when the registration data is handled improperly.*		
	Hypothesis 2	The the company might be a private body processing registration data, which might be personal data, using its CRM system, which might be a data processing system.		
	Major Premise 2	**(BDSG Section 1)** This Act shall apply to the collection, processing and use of personal data By … private bodies in so far as they process or use data by means of data processing systems or collect data for such systems …		
	Subsumption 2	**Hypothesis 2.1**	The company might be a private body	
		Major Premise 2.1	**(BDSG Section 2)** (4) "Private bodies" means natural or legal persons, companies and other private-law associations where they are not covered by sub-sections 1 to 3 above. To the extent that a private body performs sovereign public administration duties, it shall be treated as a public body for the purposes of this Act.	
		Subsumption 2.1	The company is a company which is not sovereign public administration duties.	
		Conclusion 2.1	*The company is a private body*	
		Hypothesis 2.2	Registration data, including full name and address, could be personal data	
		Major Premise 2.2	**(BDSG Section 3)** "Personal data" means any information concerning the personal or material circumstances of an identified or identifiable individual.	
		Subsumption 2.2	**Hypothesis 2.2.1**	The full name might identify an participant or might make him/her identifiable.
			Major Premise 2.2.1	**(Black's Law Dictionary)** *Identification:* Proof of identity; the proving that a person, subject, or article before the court is the very same that he or it is alleged, charged, or reputed to be; True identity is collected from a multitude of signs.
			Subsumption 2.2.1	The full name might not reveal the true identity in all cases. But it is one of the signs which can be used to derive the true identity.
			Conclusion 2.2.1	*The full name identifies an participant or makes him/her identifiable.*
			Hypothesis 2.2.2	The address might describe personal circumstances.
			Major Premise 2.2.2	**(Black's Law Dictionary)** *Personal Circumstances:* Information relating to the private aspects of a person's life.
			Subsumption 2.2.2	The information where somebody is living reveals private aspects of this person's life.
			Conclusion 2.2.2	*The address describes personal circumstances.*
			Registration data includes information about personal circumstances. And it identifies the related person or makes the person identifiable.	
		Conclusion 2.2	*Registration data, including full name and address, is personal data*	
		Hypothesis 2.3	The CRM system might be a data processing system.	
		Major Premise 2.3	**(Black's Law Dictionary)** A computer that math operations are done on. The input is changed into the output asked for by the user.	
		Subsumption 2.3	The CRM system runs on an internal data center, which can be regarded as computer. The CRM system stores the input given to it in an electronic way and allows to do computations and reporting(math operations) based on this data.	
		Conclusion 2.3	*The CRM system is a data processing system.*	
		The the company is a private body processing registration data, which identifies the participant revealing information about the participant, using its CRM system, which is runs on a computer and allows math operations.		
	Conclusion 2	*The the company is a private body processing registration data, which is personal data, using its CRM system, which is a data processing system.*		
	The company has to handle the registration data properly to protected the rights and privacy needs of the participants.			
Conclusion	*The company has to ensure that specific rights and privacy needs of the participants, who are entitled to the registration data, are preserved.*			

Table E.1.: *Full Subsumption Example*

E.2. Example Law Texts for Instantiation of Law Pattern

Table E.2 shows a complete, a referring, and a restricting dictates contained in BDSG Section 4b. Table E.3 and Table E.4[3] show the definition dictates contained in BDSG Section 3. Table E.5[4] shows several restricting dictates contained in BDSG Section 4c.

(1) *The transfer of personal data to bodies*	:	*1. in other Member States of the European Union,* *2. in other states parties to the Agreement on the European Economic Area or* *3. institutions and bodies of the European Communities*

(**Complete Dictate of Justice**)
shall be *subject to Section 15 (1), Section 16 (1) and Sections 28 to 30a in accordance with the laws and agreements applicable to such transfer, in so far as transfer is effected in connection with activities which fall in part or in their entirety within the scope of the law of the European Communities.*(**Referring Dictate of Justice**)
(2) *Sub-Section 1 shall apply mutatis mutandis to the transfer of personal data to bodies in accordance with sub-Section 1 when effected outside of activities which fall in part or in their entirety within the scope of the law of the European Communities and to the transfer of such data to other foreign, supranational or international bodies. Transfer shall not be effected in so far as the data subject has a legitimate interest in excluding transfer, in particular if an adequate level of data protection is not guaranteed at the bodies stated in the first sentence of this sub-section. The second sentence shall not apply if transfer is necessary in order to enable a public body of the Federation to perform its duties for compelling reasons of defence or to discharge supranational or international duties in the field of crisis management or conflict prevention or for humanitarian measures.*(**Restricting Dictate of Justice**)
(**Legal Consequences:**) (3) The adequacy of the afforded level of protection shall be assessed in the light of all circumstances surrounding a data transfer operation or a category of data transfer operations; particular consideration shall be given to the nature of the data, the purpose, the duration of the proposed processing operation, the country of origin, the recipient country and the legal norms, professional rules and securities measures which apply to the recipient.
(4) In the cases referred to in Section 16 (1) No. 2 above, the body transferring the data shall inform the data subject of the transfer of his/her data. This shall not apply if it can be assumed that the data subject will acquire knowledge of such transfer in another manner or if such information would jeopardize public safety or otherwise be detrimental to the Federation or a Land.
(5) Responsibility for the admissibility of the transfer shall rest with the body transferring the data.
(6) The body to which the data are transferred shall be informed of the purpose for which the data are transferred.

Table E.2.: *Complete, Referring, and Restricting Dictates Of Justice, contained in BDSG Section 4b (Bundestag der Bundesrepublik Deutschland (Lower House of German Parliament), 2009 [88])*

[3] Page 489
[4] Page 489

(1) *"Personal data" means any information concerning the personal or material circumstances of an identified or identifiable individual (the data subject).*(**Definition Dictate of Justice**)
(2) *"Automated processing" means the collection, processing or use of personal data by means of data processing systems. A non-automated filing system is any non-automated collection of personal data which is similarly structured and which can be accessed and evaluated according to specific characteristics.*(**Definition Dictate of Justice**)
(3) *"Collection" means the acquisition of data on the data subject.*(**Definition Dictate of Justice**)
(4) *"Processing" means the storage, modification, transfer, blocking and erasure of personal data. In particular cases, irrespective of the procedures applied:*

1. "storage" *means the entry, recording or preservation of personal data on a storage medium so that they can be processed or used again,*
2. "modification" *means the alteration of the substance of stored personal data,*
3. "transfer" *means the disclosure to a third party of personal data stored or obtained by means of data processing either*
   - *a) through transmission of the data to the third party or*
   - *b) through the third party inspecting or retrieving data held ready for inspection or retrieval,*
4. "blocking" *means labelling stored personal data so as to restrict their further processing or use,*
5. "erasure" *means the deletion of stored personal data.*

(**Definition Dictate of Justice**)
(5) *"Use" means any utilization of personal data other than processing.*(**Definition Dictate of Justice**)
(6) *"Rendering anonymous" means the modification of personal data so that the information concerning personal or material circumstances can no longer or only with a disproportionate amount of time, expense and labour be attributed to an identified or identifiable individual.*(**Definition Dictate of Justice**)
(6a) *"Aliasing" means replacing a person's name and other identifying characteristics with a label, in order to preclude identification of the data subject or to render such identification substantially difficult.*(**Definition Dictate of Justice**)
(7) *"Controller" means any person or body collecting, processing or using personal data on his or its own behalf or commissioning others to do the same.*(**Definition Dictate of Justice**)
(8) *"Recipient" means any person or body receiving data. "Third party" means any person or body other than the controller. This shall not include the data subject or persons and bodies commissioned to collect, process or use personal data in Germany, in another Member State of the European Union or in another state party to the Agreement on the European Economic Area.*(**Definition Dictate of Justice**)
(9) *"Special categories of personal data" means information on a person's racial or ethnic origin, political opinions, religious or philosophical convictions, union membership, health or sex life.*(**Definition Dictate of Justice**)

Table E.3.: *Definition Dictates Of Justice contained in BDSG Section 3 (Bundestag der Bundesrepublik Deutschland (Lower House of German Parliament), 2009 [88]) (1 of 2)*

(10) *"Mobile personal storage and processing media" means storage media'*

1. *which are issued to the data subject,*
2. *on which personal data can be processed automatically beyond the storage function by the issuing body or another body and,*
3. *which enable the data subject to influence this processing only by using the medium.*

(**Definition Dictate of Justice**)
(11) *"Employees" include :*

1. *employees,*
2. *persons hired for the purpose of occupational training,*
3. *persons participating in measures to integrate them into the labour market or to clarify their ability or suitability for work (rehabilitation measures),*
4. *persons employed at certified workshops for persons with a disability,*
5. *persons employed under the Youth Volunteer Service Act,*
6. *persons comparable to employees due to their economic dependence, including home-based workers and those of similar status,*
7. *applicants for employment and those whose employment has ended,*
8. *civil servants, federal judges, military personnel and persons in the alternative civilian service.*

(**Definition Dictate of Justice**)

Table E.4.: *Definition Dictates Of Justice contained in BDSG Section 3 (Bundestag der Bundesrepublik Deutschland (Lower House of German Parliament), 2009 [88]) (2 of 2)*

(1) *In connection with activities which fall in part or in their entirety within the scope of the law of the European Communities, the transfer of personal data to bodies other than those stated in Section 4b (1) above shall be admissible even if such bodies do not guarantee an adequate level of data protection, in so far as*

1. *the data subject has given his/her consent,*
2. *the transfer is necessary for the performance of a contract between the data subject and the controller or the implementation of pre-contractual measures taken in response to the data subject's request,*
3. *the transfer is necessary for the conclusion or performance of a contract which has been or is to be entered into in the interest of the data subject between the controller and a third party,*
4. *the transfer is necessary on important public interest grounds, or for the establishment, exercise or defense of legal claims,*
5. *the transfer is necessary in order to protect the vital interests of the data subject,*
6. *the transfer is made from a register which is intended to provide information to the public and which is open to consultation either by the public in general or by any person who can demonstrate a legitimate interest, to the extent that the statutory conditions are fulfilled in the particular case.*

It shall be pointed out to the recipient body that the transferred data may be processed or used only for the purpose for which they have been transferred. (**Restricting Dictate of Justice**)
(2) *Without prejudice to the first sentence of Section 1, the competent supervisory authority may authorize individual transfers or certain categories of transfers of personal data to bodies other than those stated in Section 4b (1) above, if the controller adduces adequate safeguards with respect to the protection of privacy and exercise of the corresponding rights; such safeguards may in particular result from contractual clauses or binding corporate regulations. In the case of postal and telecommunications companies, competence lies with the Federal Commissioner for Data Protection and Freedom of Information. In so far as transfer is to be effected by public bodies, the latter shall carry out the examination in accordance with the first sentence of Section 1 above.* (**Restricting Dictate of Justice**)
(**Legal Consequences:**)
(3) The Länder shall notify the Federation of the decisions made in accordance with sentence 1 of Section 2 above.

Table E.5.: *Restricting Dictates Of Justice, contained in BDSG Section 4c (Bundestag der Bundesrepublik Deutschland (Lower House of German Parliament), 2009 [88])*

E.3. Law Structure Element Hierarchies

This appendix shows full hierarchies for the BDSG. Figure E.1 shows the subject hierarchy, Figure E.2 activities hierarchy, and Figure E.3[5] the person hierarchy.

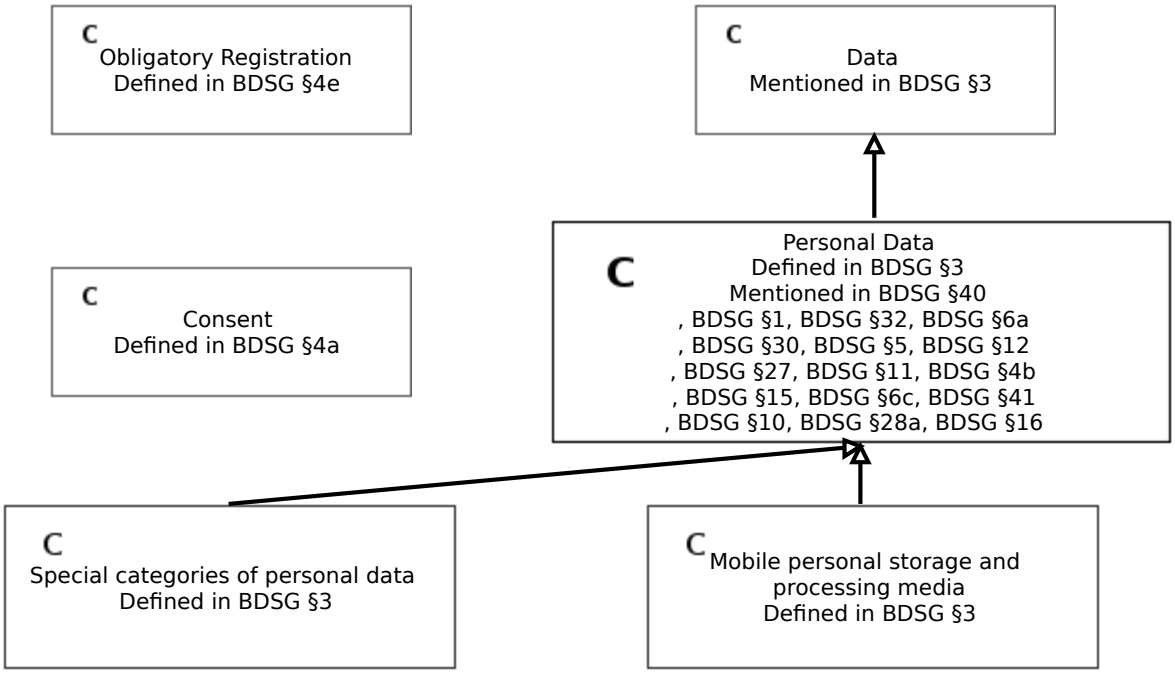


Figure E.1.: *Law Element Hierarchy for Subjects as defined by the BDSG Bundestag der Bundesrepublik Deutschland (Lower House of German Parliament) (2009 [88])*

[5] Page 492

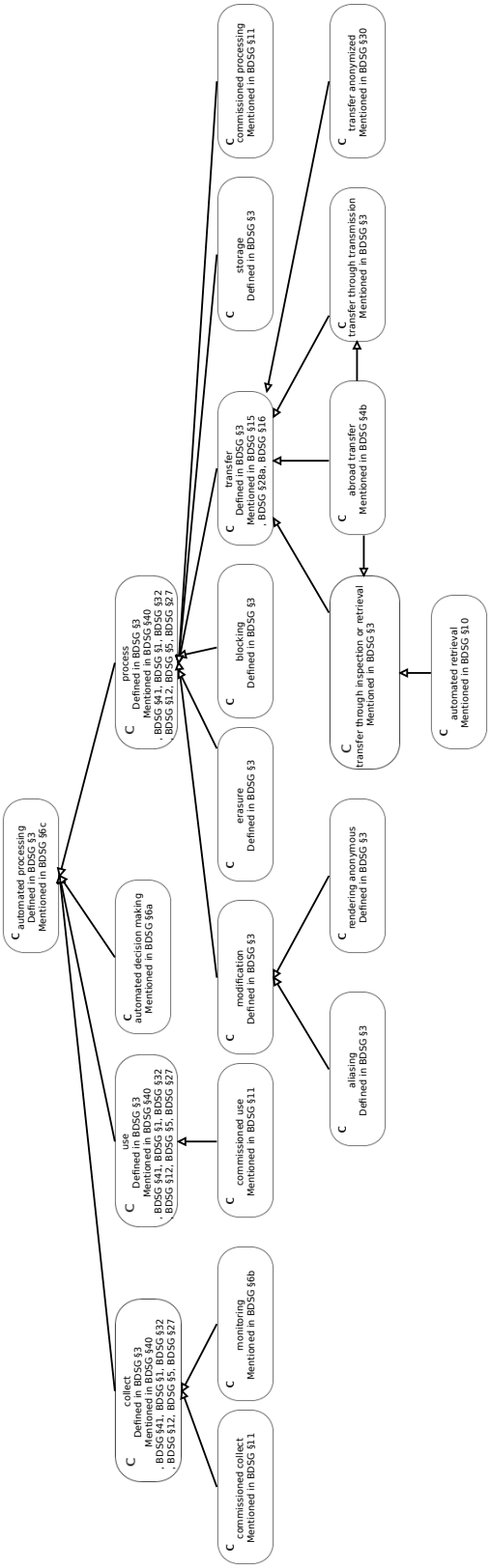


Figure E.2.: *Law Element Hierarchy for Activities as defined by the BDSG Bundestag der Bundesrepublik Deutschland (Lower House of German Parliament) (2009 [88])*

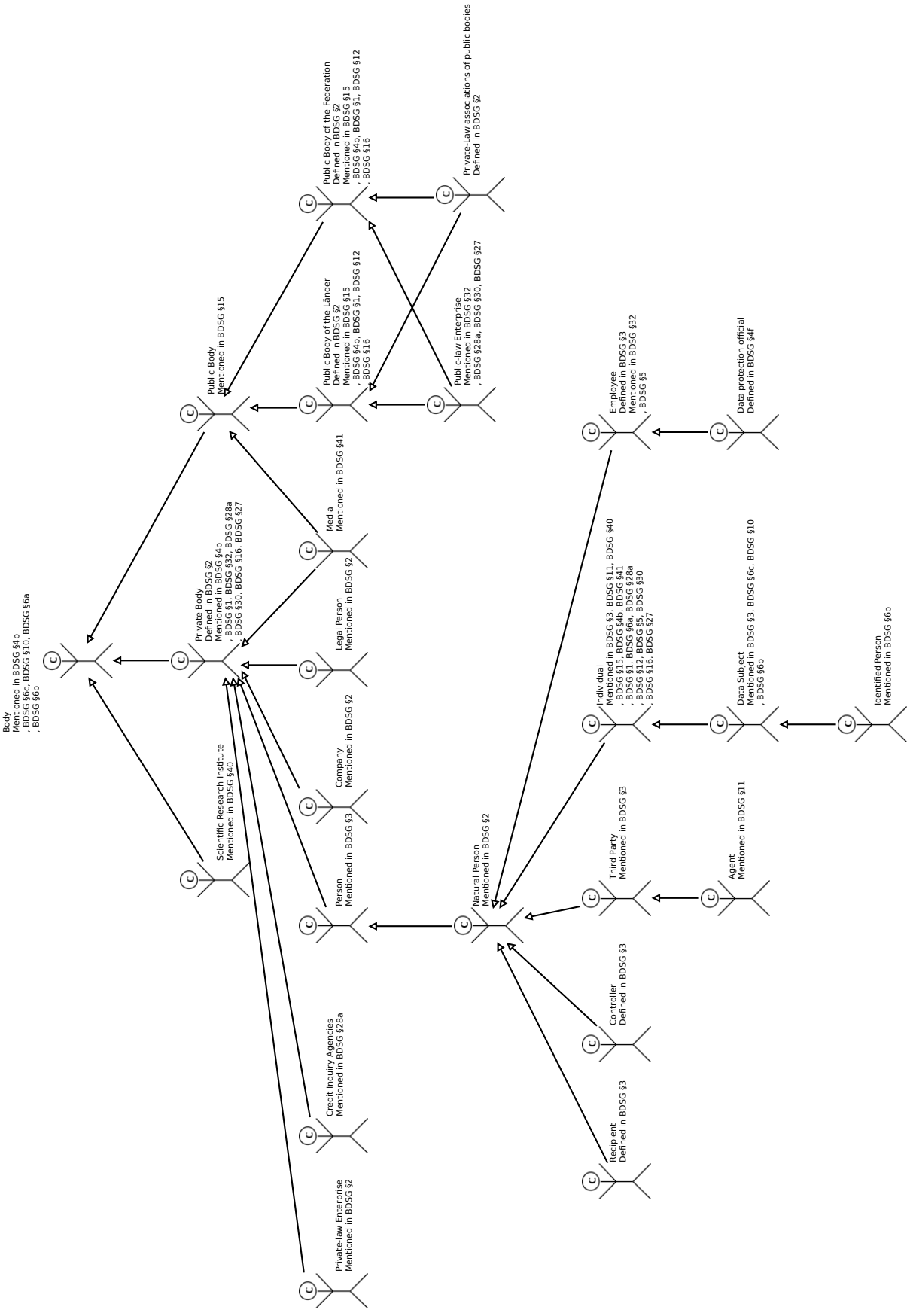


Figure E.3.: *Law Element Hierarchy for Persons as defined by the BDSG Bundestag der Bundesrepublik Deutschland (Lower House of German Parliament) (2009 [88])*

Appendix F

Cases

These chapter contains the results of applying the methods proposed in this thesis to our cases. Note that not all methods were applied for each case. Appendix F.1 contains the results regarding the media market, which is our running example. Appendix F.2[1] (Smart Grid) and Appendix F.3[2] (Voting System) contain the results regarding the cases used for validation.

F.1. Media Market

F.1.1. Context Elicitation

In the following we show the results of the context elicitation containing the information collection template instances for technical elements (Appendix F.1.1.1), the information collection template instances for indirect stakeholders (Appendix F.1.1.2[3]), as well as the information collection template instances for direct stakeholders (Appendix F.1.1.3[4]).

F.1.1.1. Information Collection Template Instances for Technical Elements

Name Payment Establishment Service

Description The Payment Establishment Service is a business service as it implements the business logic necessary to execute the payment process. within the process, the payment data provided is checked and in case it is valid the payment is executed. This requires an interaction with payment gateways which act as intermediaries for the corresponding banks.

Technological Element Relations

To	Type	Description
Service Lookup Service	relies on	From time to time the list of payment gateways has to be updated.
Payment Gateway Service	relies on	The business process part which is necessary for checking the payment data and executing the payment is outsourced to payment gateways.
Platform Service	relies on	The Payment Establishment Service runs on a Platform Service.

Table F.1.: *Template Instance for Payment Establishment Service*

[1] Page 549
[2] Page 549
[3] Page 495
[4] Page 507

Name Content Aggregation Service

Description The Content Aggregation Service is a business service as it implements the business logic necessary to support the Content Look Up process. It responds to searches requested by the customer. For fulfilling the request it spreads the request to different content providers and aggregates their responds.

Technological Element Relations

To	Type	Description
Service Lookup Service	relies on	From time to time the list of content providers has to be updated before a search is spread by the Content Aggregation Service.
Platform Service	relies on	The Content Aggregation Service runs on a Platform Service.

Table F.2.: *Template Instance for Content Aggregation Service*

Name Market Startup Service

Description The Market Startup Service handles the initialization of the Media Market once it is started. It collects the required payment gateways and content providers. It also publishes the existence of the Media Market once it is running to enable consumers to find it. The Market Startup Service is a business service as it implements some business logic according to the establishment process. Additionally, it directly interacts with the content aggregator, for example, when the content aggregator has to decide which payment gateways to take.

Technological Element Relations

To	Type	Description
Service Registry Service	relies on	To publish the existence of the media market, the Market Startup Service registers the Media Market at least one service broker using the Service Registry Service provided by this service broker.
Service Lookup Service	relies on	To build the initial set of payment gateways and content providers the Market Startup Service queries at least one service broker using the Service Lookup Service provided by this service broker.
Platform Service	relies on	The Market Startup Service runs on a Platform Service.

Table F.3.: *Template Instance for Market Startup Service*

Name Content Delivery Service

Description The Content Delivery Service is a business service as it implements the business logic necessary to support the Content Delivery process. It responds to content request issued by the customer. For fulfilling the request it collects the requested content from the according content providers and delivers it to the customer.

Technological Element Relations

To	Type	Description
Platform Service	relies on	The Content Delivery Service runs on a Platform Service.

Table F.4.: *Template Instance for Content Delivery Service*

Name Payment Gateway Service

Description The Payment Gateway Service is a business service which encapsulates parts of content payment process. It checks the validity of payment data and executes the payment.

Technological Element Relations

To	Type	Description
Platform Service	relies on	The Payment Gateway Service runs on a Platform Service.

Table F.5.: *Template Instance for Payment Gateway Service*

F.1.1.2. Information Collection Template Instances for Indirect Stakeholders

Name Customer (Organization)

Description In the first place, a Media Market customer can be every person who is in search of a certain content. As the Media Market aims at Germany as pilot market, the persons who will be able to buy content have to be citizens of Germany. A customer has to be in the legal age necessary for buying things online. And he / she has to own access to one of the provided payment options.

Motivation First of all, the customer is interested in a certain piece of media. He / she wants to possess this media to consume it. The customer wants to be sure that he / she can access his / her content at every time once it is payed. A customer also wants to pay a only reasonable price, which is, of course, as low as possible. The customer wants an easy and responsive way to search for the content and to receive the content. The customer is also interested in the security of the payment as he / she only wants to pay to correct amount of money. Additionally, no attacker should be able to obtain data which enables the attacker to impersonate the customer within any payment process.

Top Level Goals Which top level goals does the stakeholder have?
☐ **Adaptability** ☐ **Compliance** ☒ **Economy** ☐ **Efficiency**
☐ **Evolvability** ☐ **Learnability** ☐ **Maintainability** ☐ **Modularity**
☒ **Performance** ☐ **Portability** ☒ **Privacy** ☒ **Reliability**
☐ **Resilience** ☐ **Re-Usability** ☐ **Robustness** ☐ **Safety** ☐ **Scalability**
☒ **Security** ☐ **Testability** ☐ **Understandability** ☒ **Usability**

Kind

☐ **Specific** Is the stakeholder a real entity? Is the stakeholder not used to represent a group?
☐ **Representative** Is the stakeholder a real existing entity? Is this stakeholder used as proxy for a group of homogeneous stakeholders?
☒ **Group** Is the stakeholder not a real existing entity? Is this stakeholder used to describe for a group of homogeneous stakeholders?

Relation to other stakeholders

To	Description
Content Aggregator	The customer has a business relation to the content aggregator as the content aggregator is his / her direct contractual partner for the content consumption.
Bank	The customer has a business relation to the bank as the bank provides the ability to pay for the content.

Table F.6.: *Template Instance for the Organization Customer*

Name Content Aggregator (Organization)

Description The content aggregator is the one running the media market. He / she offers a platform where the customers can search for content which is provided by different content providers. Hence, the customers have a single point of access and the content providers do not have to run and maintain such a system including the payment parts. The content aggregator does not own or provide any content him- / herself. The content provider is just an intermediary.

Motivation First of all, the content aggregator has financial interests. He / she wants to generate profit out of the Media Market. Hence, he / she is interested make as much content available and attract as many customers as possible. To do so, the content aggregator needs to provide a platform which is scalable as the number of customers and content providers will grow over time. Over time also the context of the Media Market will change. Hence, the content provider is also interested in a good maintainability and evolvability. The content aggregator is interest in a great user experience to get regualar customers, which means the interaction with the media market has to be always available, easy to learn, usable, with no long waiting times, and adapted to the customers needs. Of course, also a secure payment is of interest from the content providers perspective. And the content aggregator wants to be compliant to avoid legal infringements who lead to legal actions from customers, content providers or other third parties.

Top Level Goals Which top level goals does the stakeholder have?
☐ **Adaptability** ☒ **Compliance** ☒ **Economy** ☐ **Efficiency**
☒ **Evolvability** ☒ **Learnability** ☒ **Maintainability** ☐ **Modularity**
☒ **Performance** ☐ **Portability** ☐ **Privacy** ☒ **Reliability**
☐ **Resilience** ☐ **Re-Usability** ☐ **Robustness** ☐ **Safety** ☒ **Scalability**
☒ **Security** ☐ **Testability** ☒ **Understandability** ☒ **Usability**

Kind

☒ **Specific** Is the stakeholder a real entity? Is the stakeholder not used to represent a group?
☐ **Representative** Is the stakeholder a real existing entity? Is this stakeholder used as proxy for a group of homogeneous stakeholders?
☐ **Group** Is the stakeholder not a real existing entity? Is this stakeholder used to describe for a group of homogeneous stakeholders?

Relation to other stakeholders

To	Description
Customer	The content aggregator has a contractual relation to the customer as the content he / she delivers has to be paid and the content aggregator has to assure the correct delivery.
Content Provider	The content aggregator has contracts with different content providers which regulate the content delivery and the payment modalities.
Bank	The content aggregator has to interact with different banks to execute the payment.

Table F.7.: *Template Instance for the Organization Content Aggregator*

Name Content Provider (Organization)

Description The content provider posses media items he / she is eligible to sell. The content provider runs IT systems for storing and publishing this media in a digital form. The content provider is not interested in running an own media market or he wants to improve the reach of his / her business.

Motivation The content provider wants to make profit by selling media in a digital form. He / she also wants to keep the costs for the media publishing low to increase the revenue of each sold media item. A content provider is also interested in the confidentiality of the media. Only customers who pay for a certain media shall be able to obtain and consume the media. Of course, the content provider is interested in a high availability of the Media Market as each downtime might disable a potential customer to actually by content from the content provider. Additionally, the content provider is interested in a scalable solution to serve as many customers as possible. Last but not least, the content provider is interested in learning as much as possible about his / her customers to improve his / her ability to address the customers needs in the future and gain even more revenue.

Top Level Goals Which top level goals does the stakeholder have?
☐ **Adaptability** ☒ **Compliance** ☒ **Economy** ☐ **Efficiency**
☐ **Evolvability** ☐ **Learnability** ☒ **Maintainability** ☐ **Modularity**
☐ **Performance** ☐ **Portability** ☐ **Privacy** ☒ **Reliability**
☐ **Resilience** ☐ **Re-Usability** ☐ **Robustness** ☐ **Safety** ☒ **Scalability**
☒ **Security** ☐ **Testability** ☐ **Understandability** ☐ **Usability**

Kind

☐ **Specific** Is the stakeholder a real entity? Is the stakeholder not used to represent a group?
☐ **Representative** Is the stakeholder a real existing entity? Is this stakeholder used as proxy for a group of homogeneous stakeholders?
☒ **Group** Is the stakeholder not a real existing entity? Is this stakeholder used to describe for a group of homogeneous stakeholders?

Relation to other stakeholders

To	Description
Content Aggregator	The content aggregator has a contract with the content provider which regulates the content delivery and the payment modalities.
Bank	The content provider has to have a bank account to enable payments.

Table F.8.: *Template Instance for Organization Content Provider*

Name Bank (Organization)

Description A bank offers every service regarding payments and money.

Motivation The bank not directly interested in the Media Market. But it is concerned with the security of the payment.

Top Level Goals Which top level goals does the stakeholder have?
☐ **Adaptability** ☒ **Compliance** ☒ **Economy** ☐ **Efficiency**
☐ **Evolvability** ☐ **Learnability** ☐ **Maintainability** ☐ **Modularity**
☐ **Performance** ☐ **Portability** ☐ **Privacy** ☒ **Reliability**
☐ **Resilience** ☐ **Re-Usability** ☐ **Robustness** ☐ **Safety** ☐ **Scalability**
☒ **Security** ☐ **Testability** ☐ **Understandability** ☐ **Usability**

Kind

☐ **Specific** Is the stakeholder a real entity? Is the stakeholder not used to represent a group?
☐ **Representative** Is the stakeholder a real existing entity? Is this stakeholder used as proxy for a group of homogeneous stakeholders?
☒ **Group** Is the stakeholder not a real existing entity? Is this stakeholder used to describe for a group of homogeneous stakeholders?

Relation to other stakeholders

To	Description
Content Provider	The content provider has to have a bank account to enable payments.
Content Aggregator	The content aggregator has to interact with the bank to execute the payment.
Customer	The customer has a business relation to the bank as the bank provides the ability to pay for the content.

Table F.9.: *Template Instance for Organization Bank*

Name Germany

Description In Germany several institutions form the legislator. On the federal level, the Bundestag and Bundesrat are empowered to enact regulations regarding some legal areas, while for other areas the different Landtage of the different states of Germany enact laws.

Motivation The objective of the legislator Germany is to enact regulations which regulate the daily living in way that it satisfies the personal needs of each citizen to the maximum extent without impairing the needs of other to a non-acceptable level. In general, the main objective of the legislator to ensure the best possible development of the German society.

Top Level Goals Which top level goals does the stakeholder have?
☐ **Adaptability** ☒ **Compliance** ☒ **Economy** ☐ **Efficiency**
☐ **Evolvability** ☐ **Learnability** ☐ **Maintainability** ☐ **Modularity**
☐ **Performance** ☐ **Portability** ☒ **Privacy** ☐ **Reliability**
☐ **Resilience** ☐ **Re-Usability** ☐ **Robustness** ☒ **Safety** ☐ **Scalability**
☒ **Security** ☐ **Testability** ☒ **Understandability** ☐ **Usability**

Kind

☐ **Specific** Is the stakeholder a real entity? Is the stakeholder not used to represent a group?
☒ **Representative** Is the stakeholder a real existing entity? Is this stakeholder used as proxy for a group of homogeneous stakeholders?
☐ **Group** Is the stakeholder not a real existing entity? Is this stakeholder used to describe for a group of homogeneous stakeholders?

Influence

On	Description	Severity
Customer	The laws of Germany are binding for its inhabitants	Maximum Severity
Content Aggregator	The laws of Germany are binding for every company who is selling goods or services within Germany	Maximum Severity
Content Provider	The laws of Germany are binding for every company who is selling goods or services within Germany	Maximum Severity
Bank	The laws of Germany are binding for every company who is selling goods or services within Germany	Maximum Severity

optional entries

Law candidates *(Legislator)* BDSG, HGB, TKG, TMG, GG, MARisk, Basel II, Basel III

Table F.10.: *Template Instance for Germany*

Name EU

Description In the EU several institutions form the legislator. On the EU level, the president, the EU parliament, and the ministers empowered to enact regulations. Most of those regulations have to be transformed into regulations specific for each member state by the according national legislator.

Motivation The objective of the legislator EU is to enact regulations which regulate the daily living in way that it satisfies the personal needs of each citizen to the maximum extent without impairing the needs of other to a non-acceptable level. In general, the main objective of the legislator to ensure the best possible development of the EU society.

Top Level Goals Which top level goals does the stakeholder have?
☐ **Adaptability** ☒ **Compliance** ☒ **Economy** ☐ **Efficiency**
☐ **Evolvability** ☐ **Learnability** ☐ **Maintainability** ☐ **Modularity**
☐ **Performance** ☐ **Portability** ☒ **Privacy** ☐ **Reliability**
☐ **Resilience** ☐ **Re-Usability** ☐ **Robustness** ☒ **Safety** ☐ **Scalability**
☒ **Security** ☐ **Testability** ☒ **Understandability** ☐ **Usability**

Kind

☐ **Specific** Is the stakeholder a real entity? Is the stakeholder not used to represent a group?
☒ **Representative** Is the stakeholder a real existing entity? Is this stakeholder used as proxy for a group of homogeneous stakeholders?
☐ **Group** Is the stakeholder not a real existing entity? Is this stakeholder used to describe for a group of homogeneous stakeholders?

Influence

On	Description	Severity
Customer	Some laws of EU are binding for its inhabitants, but still the national regulations are of more importance	High Severity
Content Aggregator	The laws of EU are binding for every company who is selling goods or services within the EU, but national regulations might overrule the EU ones	High Severity
Content Provider	The laws of EU are binding for every company who is selling goods or services within the EU, but national regulations might overrule the EU ones	High Severity
Bank	The laws of EU are binding for every company who is selling goods or services within the EU, but national regulations might overrule the EU ones	High Severity
Payment Gateway	The laws of EU are binding for every company who is selling goods or services within the EU, but national regulations might overrule the EU ones	High Severity
Service Broker	The laws of EU are binding for every company who is selling goods or services within the EU, but national regulations might overrule the EU ones	High Severity
Cloud Provider	The laws of EU are binding for every company who is selling goods or services within the EU, but national regulations might overrule the EU ones	High Severity

optional entries

Law candidates *(Legislator)* Directive 95/46/EC

Table F.11.: *Template Instance for the EU*

Name USA

Description In the USA several institutions form the legislator. On the federal level, the US congress, the US senate, and the US house of representatives are empowered to enact regulations regarding some legal areas, while for other areas the the congress, senate, and house of representatives of the different states of America enact laws.

Motivation The objective of the legislator US is to enact regulations which regulate the daily living in way that it satisfies the personal needs of each citizen to the maximum extent without impairing the needs of other to a non-acceptable level. In general, the main objective of the legislator to ensure the best possible development of the US society.

Top Level Goals Which top level goals does the stakeholder have?
☐ **Adaptability** ☒ **Compliance** ☒ **Economy** ☐ **Efficiency**
☐ **Evolvability** ☐ **Learnability** ☐ **Maintainability** ☐ **Modularity**
☐ **Performance** ☐ **Portability** ☒ **Privacy** ☐ **Reliability**
☐ **Resilience** ☐ **Re-Usability** ☐ **Robustness** ☒ **Safety** ☐ **Scalability**
☒ **Security** ☐ **Testability** ☒ **Understandability** ☐ **Usability**

Kind

☐ **Specific** Is the stakeholder a real entity? Is the stakeholder not used to represent a group?
☒ **Representative** Is the stakeholder a real existing entity? Is this stakeholder used as proxy for a group of homogeneous stakeholders?
☐ **Group** Is the stakeholder not a real existing entity? Is this stakeholder used to describe for a group of homogeneous stakeholders?

Influence

On	Description	Severity
Content Provider	The laws of the US are binding for every company who is selling goods or services within the US	Maximum Severity
Bank	The laws of the US are binding for every company who is selling goods or services within the US	Maximum Severity
Payment Gateway	The laws of US are binding for every company who is selling goods or services within the US	Maximum Severity
Cloud Provider	The laws of US are binding for every company who is selling goods or services within the US	Maximum Severity

optional entries

Law candidates *(Legislator)* Patriot Act, SOX

Table F.12.: *Template Instance for USA*

Name Media

Description The media domain is formed by all companies which plan, produce, and sell media content for information and entertainment purposes. The most important parts of the media domain are print media, audiovisual media, and online media.

Motivation The organizations of the media domain are especially concerned with protecting copy rights and the economic interests of media companies.

Top Level Goals Which top level goals does the stakeholder have?
☐ **Adaptability** ☒ **Compliance** ☒ **Economy** ☐ **Efficiency**
☐ **Evolvability** ☐ **Learnability** ☐ **Maintainability** ☐ **Modularity**
☐ **Performance** ☐ **Portability** ☐ **Privacy** ☐ **Reliability**
☐ **Resilience** ☐ **Re-Usability** ☐ **Robustness** ☐ **Safety** ☐ **Scalability**
☐ **Security** ☐ **Testability** ☐ **Understandability** ☐ **Usability**

Kind

☐ **Specific** Is the stakeholder a real entity? Is the stakeholder not used to represent a group?
☐ **Representative** Is the stakeholder a real existing entity? Is this stakeholder used as proxy for a group of homogeneous stakeholders?
☒ **Group** Is the stakeholder not a real existing entity? Is this stakeholder used to describe for a group of homogeneous stakeholders?

Influence

On	Description	Severity
Content Aggregator	The media domain as such can only provide guidelines, best practices, recommendations, and so forth. But neglecting such things published by organizations of the domain might be punished by other companies in the domain	low to medium
Content Provider	The media domain as such can only provide guidelines, best practices, recommendations, and so forth. But neglecting such things published by organizations of the domain might be punished by other companies in the domain	low to medium

Table F.13.: *Template Instance for Media*

Name GEMA

Description The GEMA is a German public body of the German state, which is empowered to enforce the copyright administration act of Germany, which is specific to the media domain. The GEMA is responsible for audio and audiovisual media.

Motivation The organizations of the media domain are especially concerned with protecting copy rights and the economic interests of media producers.

Top Level Goals Which top level goals does the stakeholder have?
☐ **Adaptability** ☒ **Compliance** ☒ **Economy** ☐ **Efficiency**
☐ **Evolvability** ☐ **Learnability** ☐ **Maintainability** ☐ **Modularity**
☐ **Performance** ☐ **Portability** ☐ **Privacy** ☐ **Reliability**
☐ **Resilience** ☐ **Re-Usability** ☐ **Robustness** ☐ **Safety** ☐ **Scalability**
☐ **Security** ☐ **Testability** ☐ **Understandability** ☐ **Usability**

Kind

☒ **Specific** Is the stakeholder a real entity? Is the stakeholder not used to represent a group?
☐ **Representative** Is the stakeholder a real existing entity? Is this stakeholder used as proxy for a group of homogeneous stakeholders?
☐ **Group** Is the stakeholder not a real existing entity? Is this stakeholder used to describe for a group of homogeneous stakeholders?

Influence

On	Description	Severity
Content Aggregator	As the GEMA is operating on the basis of the copyright administration act, the Media Market is forced to collaborate and follow the rules of the GEMA	Maximum
Content Providers	As the GEMA is operating on the basis of the copyright administration act, the Content Providers is forced to collaborate and follow the rules of the GEMA	Maximum

optional entries

Domain-specific regulations *(Domain)* copyright administration act.

Table F.14.: *Template Instance for GEMA*

Name VGWORT

Description The VGWORT is a German public body of the German state, which is empowered to enforce the copyright administration act of Germany, which is specific to the media domain. The GEMA is responsible for audio and audiovisual media.

Motivation The organizations of the media domain are especially concerned with protecting copy rights and the economic interests of media producers.

Top Level Goals Which top level goals does the stakeholder have?
☐ **Adaptability** ☒ **Compliance** ☒ **Economy** ☐ **Efficiency**
☐ **Evolvability** ☐ **Learnability** ☐ **Maintainability** ☐ **Modularity**
☐ **Performance** ☐ **Portability** ☐ **Privacy** ☐ **Reliability**
☐ **Resilience** ☐ **Re-Usability** ☐ **Robustness** ☐ **Safety** ☐ **Scalability**
☐ **Security** ☐ **Testability** ☐ **Understandability** ☐ **Usability**

Kind
☒ **Specific** Is the stakeholder a real entity? Is the stakeholder not used to represent a group?
☐ **Representative** Is the stakeholder a real existing entity? Is this stakeholder used as proxy for a group of homogeneous stakeholders?
☐ **Group** Is the stakeholder not a real existing entity? Is this stakeholder used to describe for a group of homogeneous stakeholders?

Influence

On	Description	Severity
Content Aggregator	As the VGWORT is operating on the basis of the copyright administration act, the Media Market is forced to collaborate and follow the rules of the VGWORT	Maximum
Content Providers	As the VGWORT is operating on the basis of the copyright administration act, the Content Providers is forced to collaborate and follow the rules of the VGWORT	Maximum

optional entries

Assets *(Asset Provider)*

Asset	Description	Provided To
Media Content	The media which is sold by the content provider, and for which somebody else has the copyright	Content Provider
Media Content	The media which is sold by the Media Market, and for which somebody else has the copyright	Content Aggregator

Table F.15.: *Template Instance for VGWORT*

Name Content Owner

Description The content owner is the one who has the copy rights for at least one piece of media sold using the media market.

Motivation The content owner is especially concerned with protecting his / her copy rights and the economic interests.

Top Level Goals Which top level goals does the stakeholder have?
☐ **Adaptability** ☒ **Compliance** ☒ **Economy** ☐ **Efficiency**
☐ **Evolvability** ☐ **Learnability** ☐ **Maintainability** ☐ **Modularity**
☐ **Performance** ☐ **Portability** ☐ **Privacy** ☐ **Reliability**
☐ **Resilience** ☐ **Re-Usability** ☐ **Robustness** ☐ **Safety** ☐ **Scalability**
☐ **Security** ☐ **Testability** ☐ **Understandability** ☐ **Usability**

Kind

☐ **Specific** Is the stakeholder a real entity? Is the stakeholder not used to represent a group?
☐ **Representative** Is the stakeholder a real existing entity? Is this stakeholder used as proxy for a group of homogeneous stakeholders?
☒ **Group** Is the stakeholder not a real existing entity? Is this stakeholder used to describe for a group of homogeneous stakeholders?

Influence

On	Description	Severity
Content Aggregator	A content owner has several copy rights which should not be infringed by the Media Market	High
Content Provider	A content owner has several copy rights which should not be infringed by the Content Provider	High

optional entries

Assets *(Asset Provider)*

Asset	Description	Provided To
Media Content	The media which is sold by the content provider, and for which the content owner has the copyright	Content Provider
Media Content	The media which is sold by the Media Market, and for which the content owner else has the copyright	Content Aggregator

Table F.16.: *Template Instance for Content Owner*

Name Finance

Description The finance domain is formed by all companies which are concerned with offering financial products or services related to financial products. The most important parts of the finance domain are banks, credit institutions, and insurances.

Motivation The organizations of the finance domain are especially concerned with protecting the stability of the (international) financial transaction systems and the economic interests of finance companies.

Top Level Goals Which top level goals does the stakeholder have?
☐ **Adaptability** ☐ **Compliance** ☒ **Economy** ☐ **Efficiency**
☐ **Evolvability** ☐ **Learnability** ☐ **Maintainability** ☐ **Modularity**
☐ **Performance** ☐ **Portability** ☐ **Privacy** ☐ **Reliability**
☒ **Resilience** ☐ **Re-Usability** ☒ **Robustness** ☐ **Safety** ☐ **Scalability**
☒ **Security** ☐ **Testability** ☐ **Understandability** ☐ **Usability**

Kind

☐ **Specific** Is the stakeholder a real entity? Is the stakeholder not used to represent a group?
☐ **Representative** Is the stakeholder a real existing entity? Is this stakeholder used as proxy for a group of homogeneous stakeholders?
☒ **Group** Is the stakeholder not a real existing entity? Is this stakeholder used to describe for a group of homogeneous stakeholders?

Influence

On	Description	Severity
Bank	The finance domain as such can only provide guidelines, best practices, recommendations, and so forth. But neglecting such things published by organizations of the domain can end up in an exclusion of the (international) transaction system	high

Table F.17.: *Template Instance for Finance*

F.1.1.3. Information Collection Template Instances for Direct Stakeholders

Name Customer (Direct Stakeholder)

Description In the first place, a Media Market customer can be every person who is in search of a certain content. As the Media Market aims at Germany as pilot market, the persons who will be able to buy content have to be citizens of Germany. A customer has to be in the legal age necessary for buying things online. And he / she has to own access to one of the provided payment options.

Motivation First of all, the customer is interested in a certain piece of media. He / she wants to possess this media to consume it. The customer wants to be sure that he / she can access his / her content at every time once it is payed. A customer also wants to pay a only reasonable price, which is, of course, as low as possible. The customer wants an easy and responsive way to search for the content and to receive the content. The customer is also interested in the security of the payment as he / she only wants to pay to correct amount of money. Additionally, no attacker should be able to obtain data which enables the attacker to impersonate the customer within any payment process.

Top Level Goals Which top level goals does the stakeholder have?
☐ **Adaptability** ☐ **Compliance** ☒ **Economy** ☐ **Efficiency**
☐ **Evolvability** ☐ **Learnability** ☐ **Maintainability** ☐ **Modularity**
☒ **Performance** ☐ **Portability** ☒ **Privacy** ☒ **Reliability**
☐ **Resilience** ☐ **Re-Usability** ☐ **Robustness** ☐ **Safety** ☐ **Scalability**
☒ **Security** ☐ **Testability** ☐ **Understandability** ☒ **Usability**

Kind

☐ **Specific** Is the stakeholder a real entity? Is the stakeholder not used to represent a group?
☐ **Representative** Is the stakeholder a real existing entity? Is this stakeholder used as proxy for a group of homogeneous stakeholders?
☒ **Group** Is the stakeholder not a real existing entity? Is this stakeholder used to describe for a group of homogeneous stakeholders?

optional entries

Takes Part In *(Process Actor)*

Process	**Activity**
Content Look Up	request content list, browse list
Content Payment	provide payment data
Content Delivery	select content, request content, abort, save content, show content

Table F.18.: *Template Instance for the Direct Stakeholder Customer*

Name Bank (Direct Stakeholder)

Description A bank offers every service regarding payments and money.

Motivation The bank not directly interested in the Media Market. But it is concerned with the security of the payment.

Top Level Goals Which top level goals does the stakeholder have?
☐ **Adaptability** ☒ **Compliance** ☒ **Economy** ☐ **Efficiency**
☐ **Evolvability** ☐ **Learnability** ☐ **Maintainability** ☐ **Modularity**
☐ **Performance** ☐ **Portability** ☐ **Privacy** ☒ **Reliability**
☐ **Resilience** ☐ **Re-Usability** ☐ **Robustness** ☐ **Safety** ☐ **Scalability**
☒ **Security** ☐ **Testability** ☐ **Understandability** ☐ **Usability**

Kind
☐ **Specific** Is the stakeholder a real entity? Is the stakeholder not used to represent a group?
☐ **Representative** Is the stakeholder a real existing entity? Is this stakeholder used as proxy for a group of homogeneous stakeholders?
☒ **Group** Is the stakeholder not a real existing entity? Is this stakeholder used to describe for a group of homogeneous stakeholders?

Takes Part In *(Process Actor)*

Process	**Activity**
Content Payment	check payment data, reject payment, accept payment, initiate payment, pay

Table F.19.: *Template Instance for Direct Stakeholder Bank*

Name Cloud Provider

Description Runs the platform service

Motivation Offering the platform service is the business of the payment gateway provider.

Top Level Goals Which top level goals does the stakeholder have?
☐ **Adaptability** ☐ **Compliance** ☒ **Economy** ☐ **Efficiency**
☐ **Evolvability** ☐ **Learnability** ☐ **Maintainability** ☐ **Modularity**
☒ **Performance** ☐ **Portability** ☐ **Privacy** ☒ **Reliability**
☒ **Resilience** ☐ **Re-Usability** ☒ **Robustness** ☐ **Safety** ☒ **Scalability**
☒ **Security** ☐ **Testability** ☐ **Understandability** ☐ **Usability**

Kind
☐ **Specific** Is the stakeholder a real entity? Is the stakeholder not used to represent a group?
☐ **Representative** Is the stakeholder a real existing entity? Is this stakeholder used as proxy for a group of homogeneous stakeholders?
☒ **Group** Is the stakeholder not a real existing entity? Is this stakeholder used to describe for a group of homogeneous stakeholders?

Provides *(* Provider)*

Provides	**Description**
Platform Service	One can use the platform service to run his/her own services

Table F.20.: *Template Instance for Direct Stakeholder Cloud Provider*

Name Content Provider (Direct Stakeholder)

Description The content provider posses media items he / she is eligible to sell. The content provider runs IT systems for storing and publishing this media in a digital form. The content provider is not interested in running an own media market or he wants to improve the reach of his / her business.

Motivation The content provider wants to make profit by selling media in a digital form. He / she also wants to keep the costs for the media publishing low to increase the revenue of each sold media item. A content provider is also interested in the confidentiality of the media. Only customers who pay for a certain media shall be able to obtain and consume the media. Of course, the content provider is interested in a high availability of the Media Market as each downtime might disable a potential customer to actually by content from the content provider. Additionally, the content provider is interested in a scalable solution to serve as many customers as possible. Last but not least, the content provider is interested in learning as much as possible about his / her customers to improve his / her ability to address the customers needs in the future and gain even more revenue.

Top Level Goals Which top level goals does the stakeholder have?
☐ **Adaptability** ☒ **Compliance** ☒ **Economy** ☐ **Efficiency**
☐ **Evolvability** ☐ **Learnability** ☒ **Maintainability** ☐ **Modularity**
☐ **Performance** ☐ **Portability** ☐ **Privacy** ☒ **Reliability**
☐ **Resilience** ☐ **Re-Usability** ☐ **Robustness** ☐ **Safety** ☒ **Scalability**
☒ **Security** ☐ **Testability** ☐ **Understandability** ☐ **Usability**

Kind
☐ **Specific** Is the stakeholder a real entity? Is the stakeholder not used to represent a group?
☐ **Representative** Is the stakeholder a real existing entity? Is this stakeholder used as proxy for a group of homogeneous stakeholders?
☒ **Group** Is the stakeholder not a real existing entity? Is this stakeholder used to describe for a group of homogeneous stakeholders?

Takes Part In *(Process Actor)*

Process	**Activity**
Content Look Up	search for matching content, summarize information
Content Delivery	deliver content

Table F.21.: *Template Instance for Direct Stakeholder Content Provider*

Name Administrator

Description The administrator is responsible for setting up the system and controlling the technical aspects of the Media Market.

Motivation The administrator is payed for doing his / her job and wants to conduct his / her job as convenient as possible.

Top Level Goals Which top level goals does the stakeholder have?
☐ **Adaptability** ☐ **Compliance** ☐ **Economy** ☐ **Efficiency**
☐ **Evolvability** ☒ **Learnability** ☒ **Maintainability** ☐ **Modularity**
☒ **Performance** ☐ **Portability** ☐ **Privacy** ☒ **Reliability**
☐ **Resilience** ☒ **Re-Usability** ☒ **Robustness** ☐ **Safety** ☒ **Scalability**
☒ **Security** ☒ **Testability** ☒ **Understandability** ☒ **Usability**

Kind
☐ **Specific** Is the stakeholder a real entity? Is the stakeholder not used to represent a group?
☐ **Representative** Is the stakeholder a real existing entity? Is this stakeholder used as proxy for a group of homogeneous stakeholders?
☒ **Group** Is the stakeholder not a real existing entity? Is this stakeholder used to describe for a group of homogeneous stakeholders?

Takes Part In *(Process Actor)*

Process	**Activity**
Establishment & Maintenance	

Table F.22.: *Template Instance for Direct Stakeholder Administrator*

Name Accounter

Description The accounter is responsible for controlling the financial aspects such as billing and so forth.

Motivation The accounter is payed for doing his / her job and wants to conduct his / her job as convenient as possible.

Top Level Goals Which top level goals does the stakeholder have?
☐ **Adaptability** ☐ **Compliance** ☐ **Economy** ☐ **Efficiency**
☐ **Evolvability** ☒ **Learnability** ☐ **Maintainability** ☐ **Modularity**
☒ **Performance** ☐ **Portability** ☐ **Privacy** ☒ **Reliability**
☐ **Resilience** ☐ **Re-Usability** ☒ **Robustness** ☐ **Safety** ☐ **Scalability**
☐ **Security** ☐ **Testability** ☒ **Understandability** ☒ **Usability**

Kind
☐ **Specific** Is the stakeholder a real entity? Is the stakeholder not used to represent a group?
☐ **Representative** Is the stakeholder a real existing entity? Is this stakeholder used as proxy for a group of homogeneous stakeholders?
☒ **Group** Is the stakeholder not a real existing entity? Is this stakeholder used to describe for a group of homogeneous stakeholders?

Takes Part In *(Process Actor)*

Process	**Activity**
Establishment & Maintenance	

Table F.23.: *Template Instance for Direct Stakeholder Accounter*

Name Payment Gateway Provider

Description Runs the payment gateway service

Motivation Offering the payment gateway is the business of the payment gateway provider.

Top Level Goals Which top level goals does the stakeholder have?

☐ **Adaptability** ☒ **Compliance** ☒ **Economy** ☐ **Efficiency**
☐ **Evolvability** ☐ **Learnability** ☐ **Maintainability** ☐ **Modularity**
☒ **Performance** ☐ **Portability** ☐ **Privacy** ☒ **Reliability**
☒ **Resilience** ☐ **Re-Usability** ☒ **Robustness** ☐ **Safety** ☒ **Scalability**
☒ **Security** ☐ **Testability** ☐ **Understandability** ☐ **Usability**

Kind

☐ **Specific** Is the stakeholder a real entity? Is the stakeholder not used to represent a group?
☐ **Representative** Is the stakeholder a real existing entity? Is this stakeholder used as proxy for a group of homogeneous stakeholders?
☒ **Group** Is the stakeholder not a real existing entity? Is this stakeholder used to describe for a group of homogeneous stakeholders?

Provides *(* Provider)*

Provides	Description
Payment Gateway Service	Provides checking of payment data and the initiation of the payment

Table F.24.: *Template Instance for Direct Stakeholder Payment Gateway Provider*

Name Service Broker

Description Runs the service registry service and the service lookup service.

Motivation Offering these services is the business of the service broker.

Top Level Goals Which top level goals does the stakeholder have?

☐ **Adaptability** ☒ **Compliance** ☒ **Economy** ☐ **Efficiency**
☐ **Evolvability** ☐ **Learnability** ☐ **Maintainability** ☐ **Modularity**
☒ **Performance** ☐ **Portability** ☐ **Privacy** ☒ **Reliability**
☐ **Resilience** ☐ **Re-Usability** ☒ **Robustness** ☐ **Safety** ☒ **Scalability**
☒ **Security** ☐ **Testability** ☐ **Understandability** ☐ **Usability**

Kind

☐ **Specific** Is the stakeholder a real entity? Is the stakeholder not used to represent a group?
☐ **Representative** Is the stakeholder a real existing entity? Is this stakeholder used as proxy for a group of homogeneous stakeholders?
☒ **Group** Is the stakeholder not a real existing entity? Is this stakeholder used to describe for a group of homogeneous stakeholders?

Provides *(* Provider)*

Provides	Description
Service Registry Service	One can register his/her service using the service registry service.
Service Lookup Service	One can search for different services using the service lookup service.

Table F.25.: *Template Instance for Direct Stakeholder Service Broke*

F.1.2. Goals

In this appendix, we show the full goal trees for the content aggregator (Figure F.1), content provider(Figure F.2[5]), customer (Figure F.3[6]), and payment gateway provider / service broker / bank (Figure F.4[7]). Note that we assume that the goals of the latter ones are quite the same regarding the Media Market. Figure F.5[8] shows the relations between goals of different stakeholders.

[5]Page 514
[6]Page 515
[7]Page 516
[8]Page 517

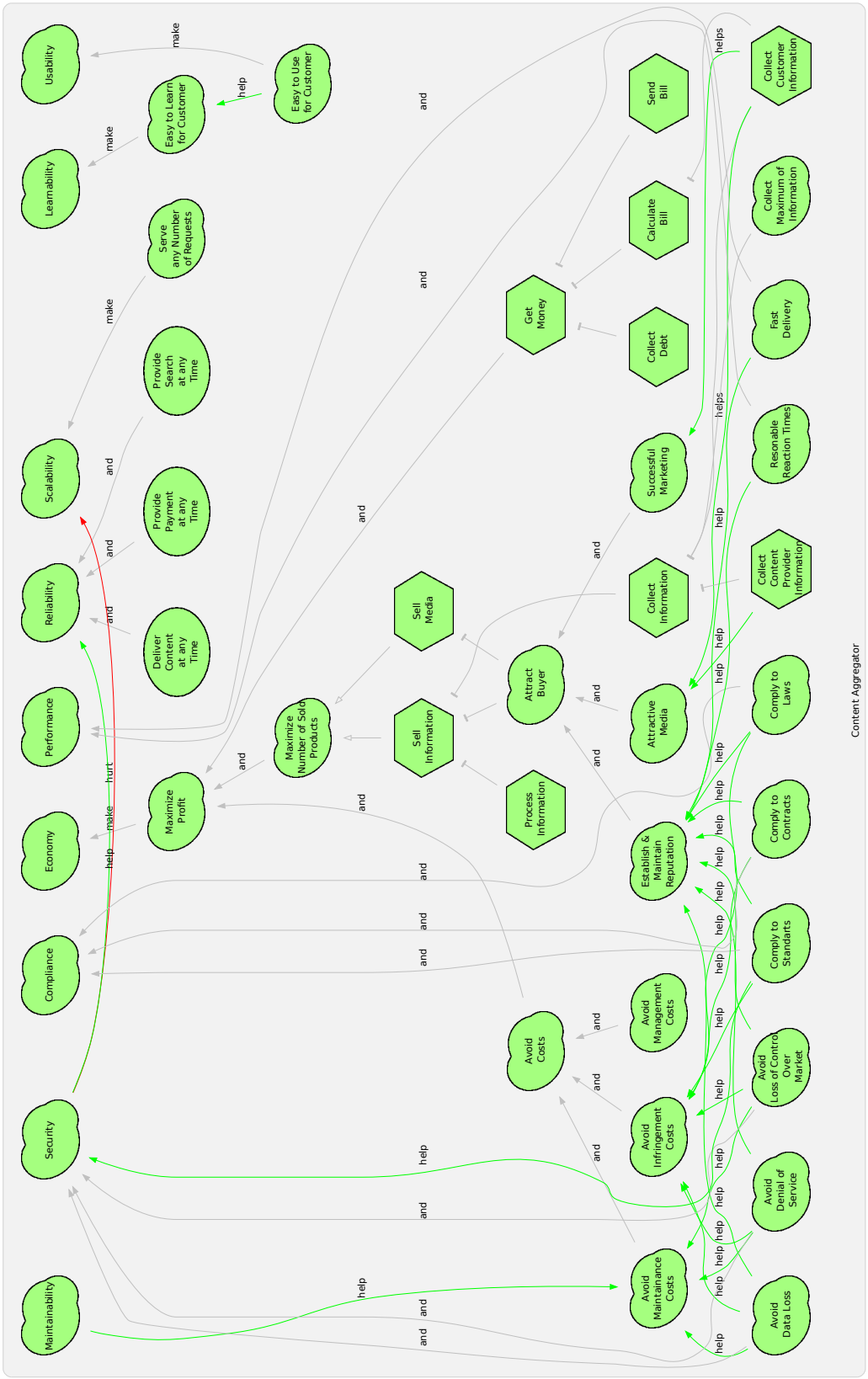


Figure F.1.: *Goal Tree for Content Aggregator*

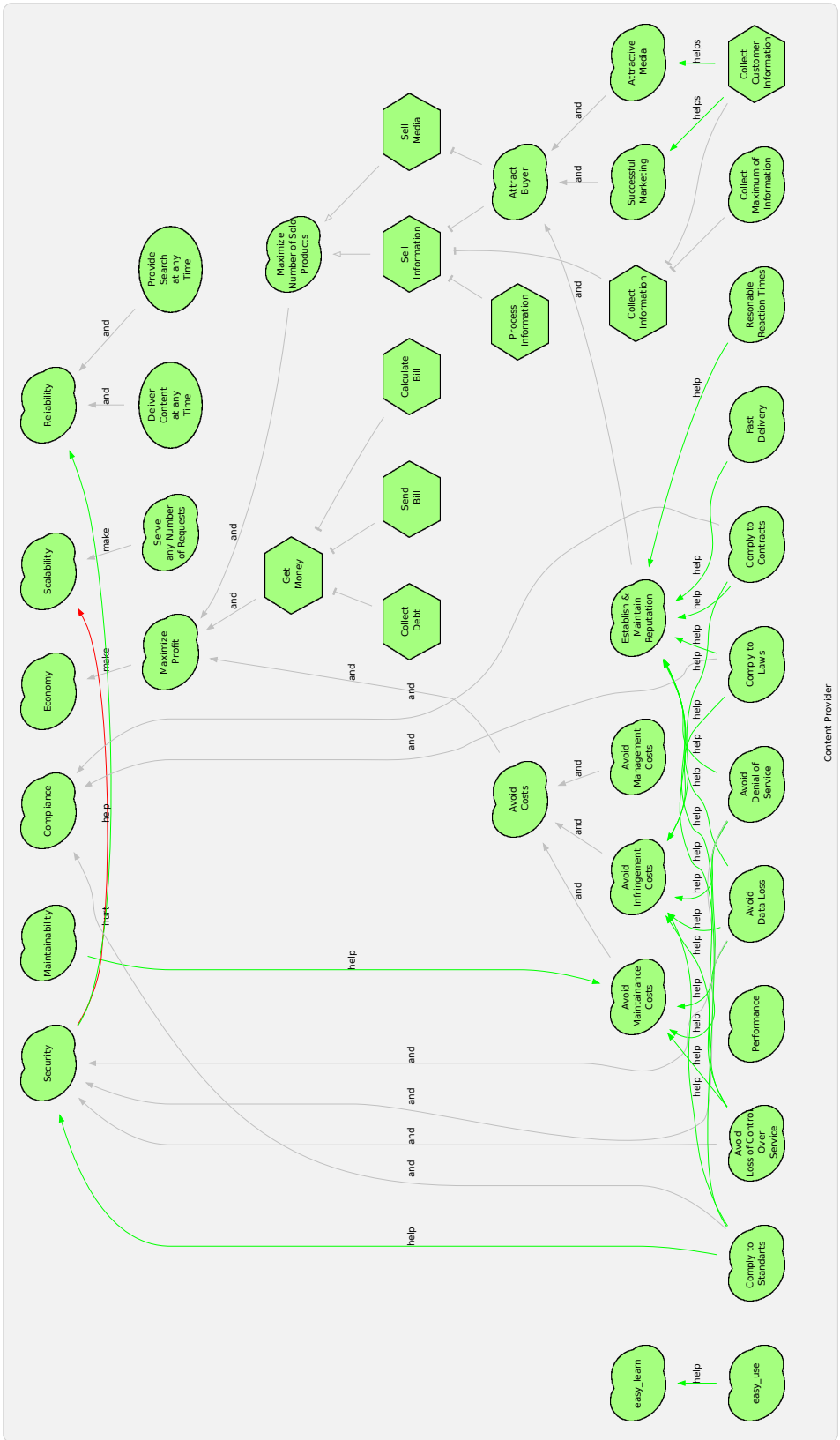


Figure F.2.: *Goal Tree for Content Provider*

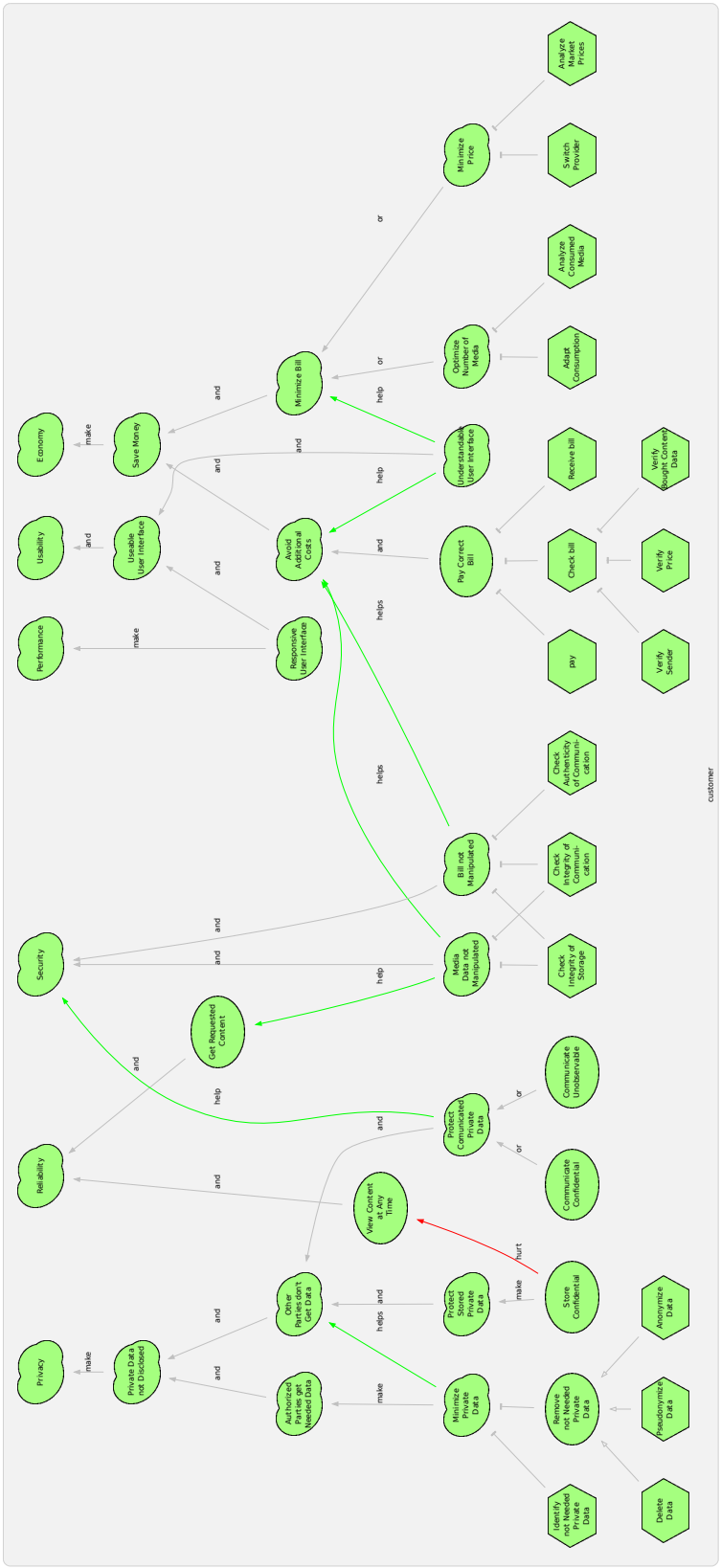


Figure F.3.: *Goal Tree for Customer*

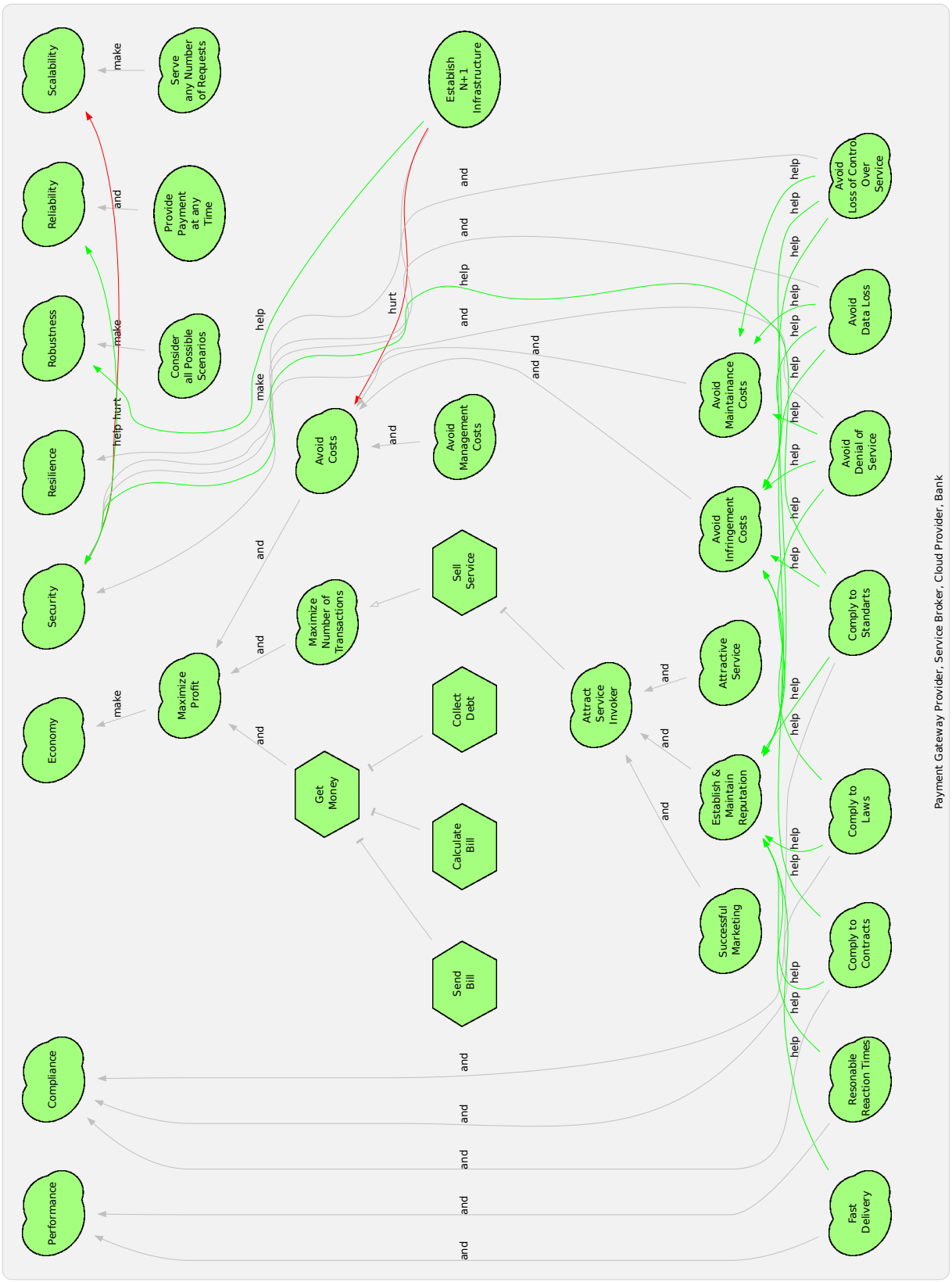


Figure F.4.: *Goal Tree for Payment Gateway Provider / Service Broker / Bank / Cloud Provider*

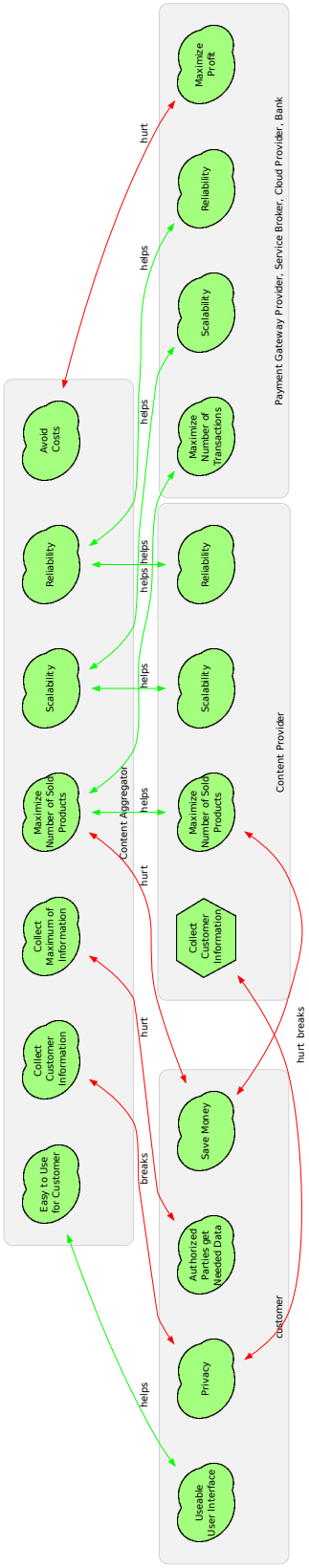


Figure F.5.: *Goal Relations Between Stakeholders*

F.1.3. Processes

In this appendix we show the results of the process elicitation. Appendix F.1.3.1 shows the EPCs, and Appendix F.1.3.2[9] the according FADs.

F.1.3.1. EPC

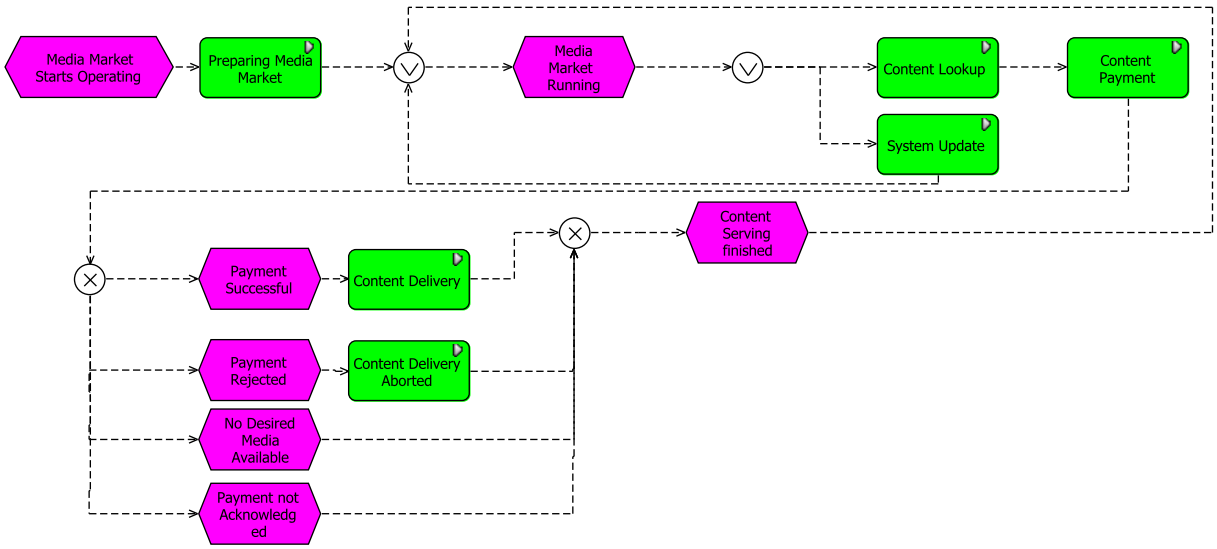


Figure F.6.: *Overall Process Media Market*

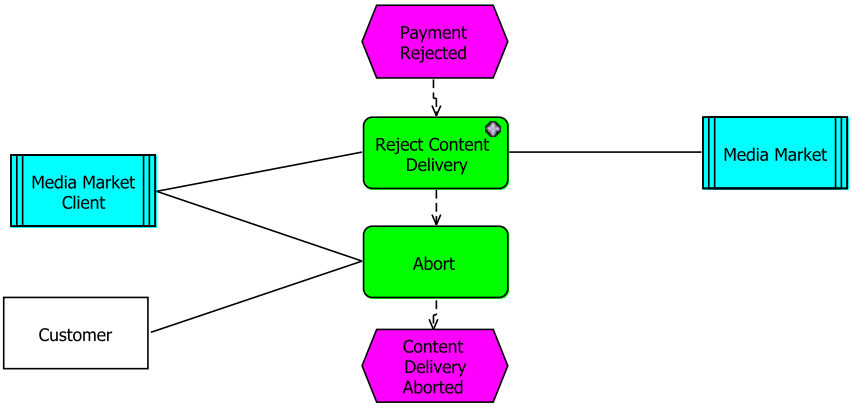


Figure F.7.: *Content Delivery Rejected*

[9]Page 521

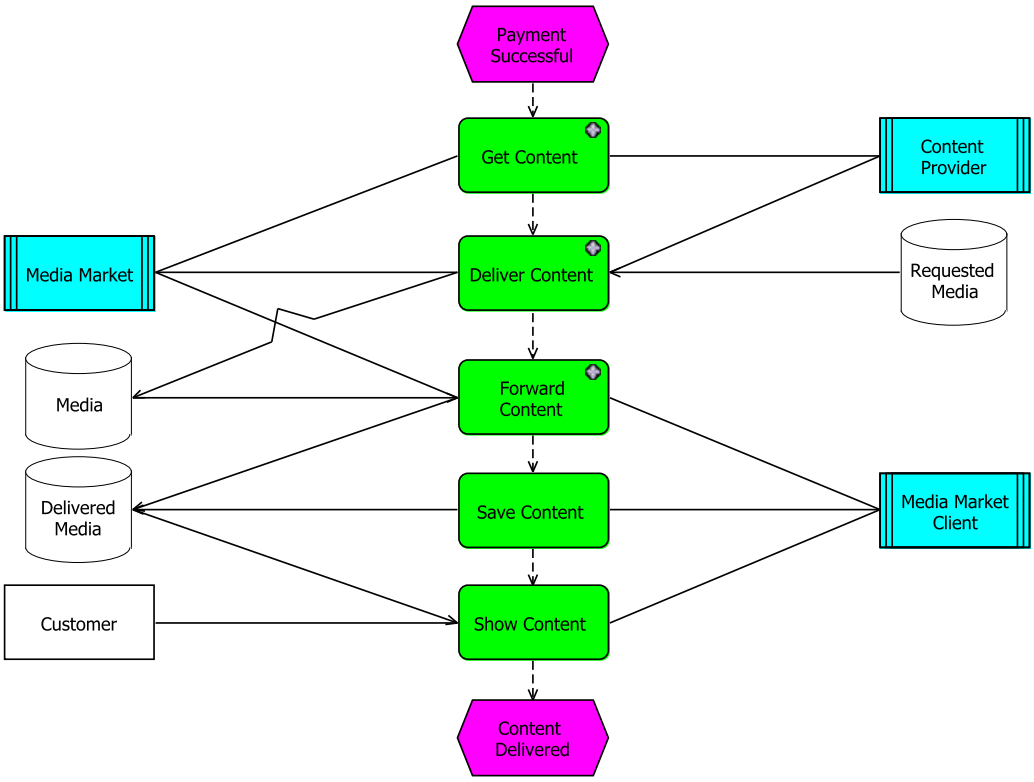


Figure F.8.: *Content Delivery*

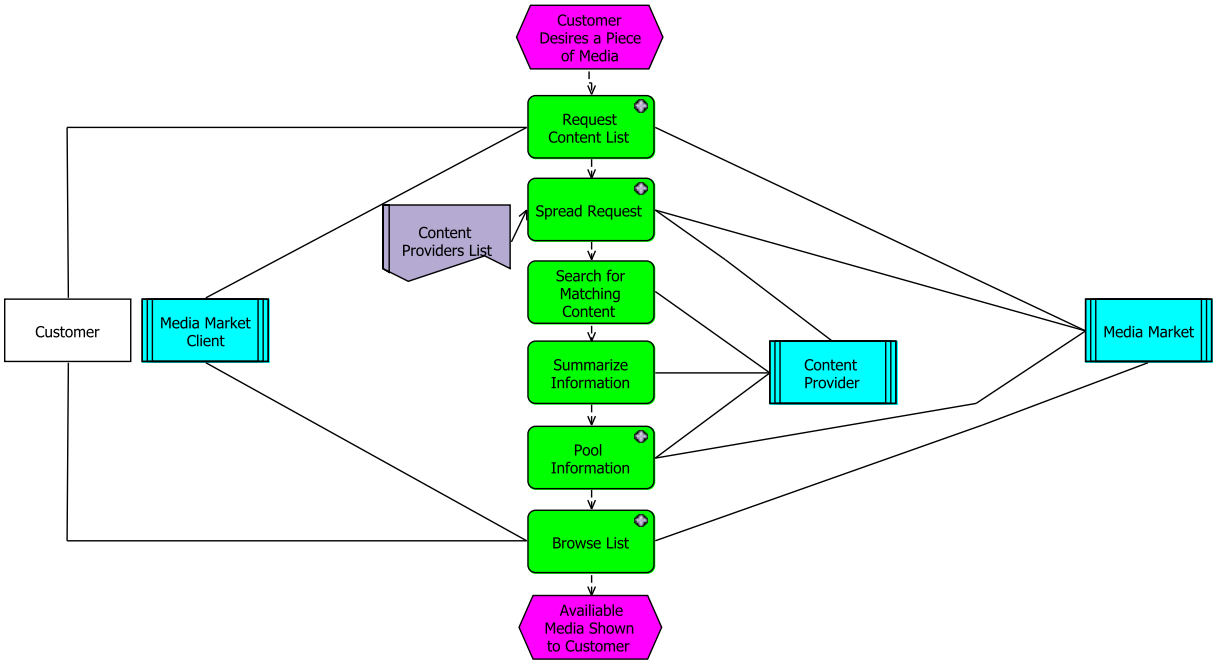


Figure F.9.: *Content Lookup*

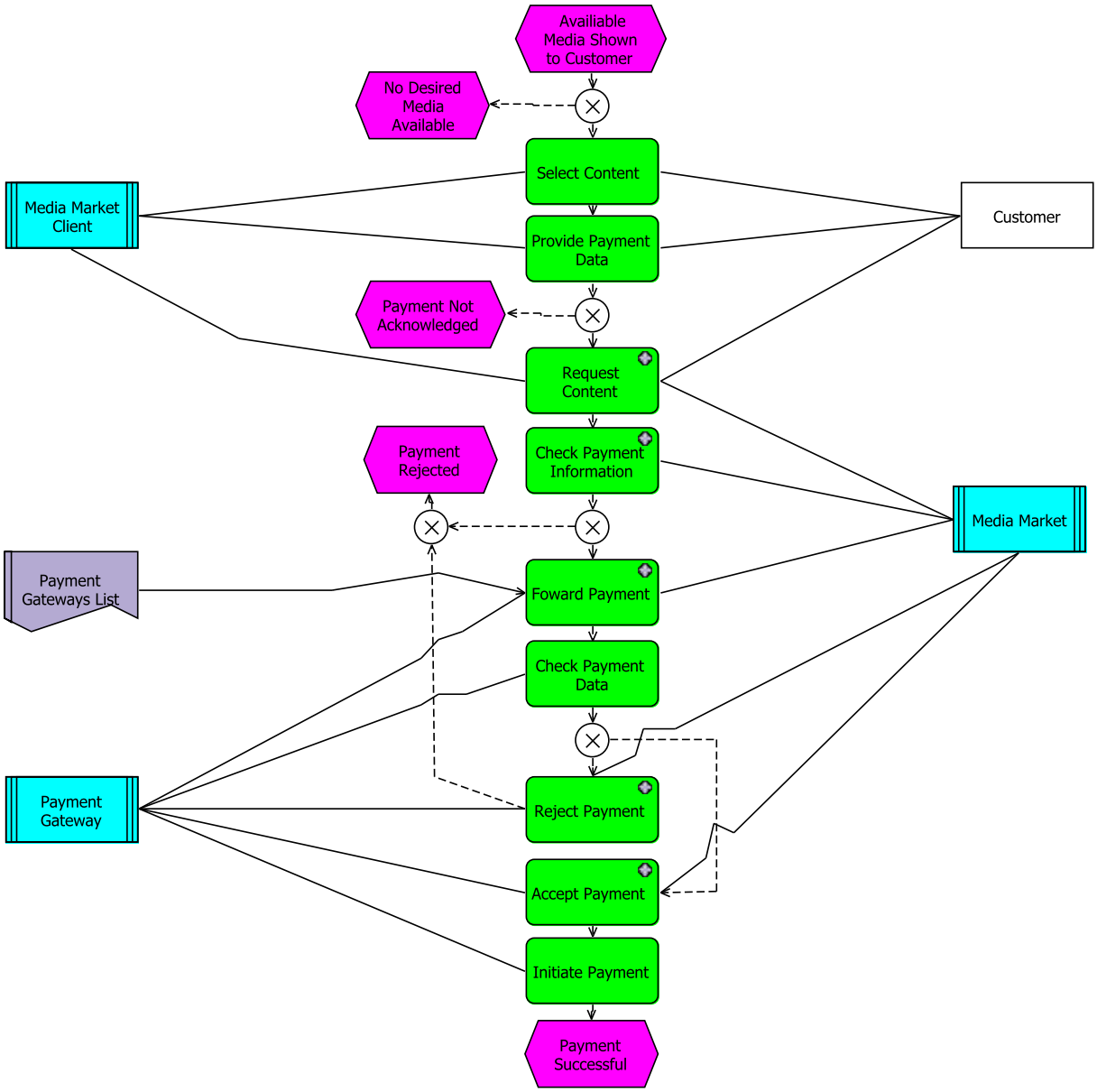


Figure F.10.: *Content Payment*

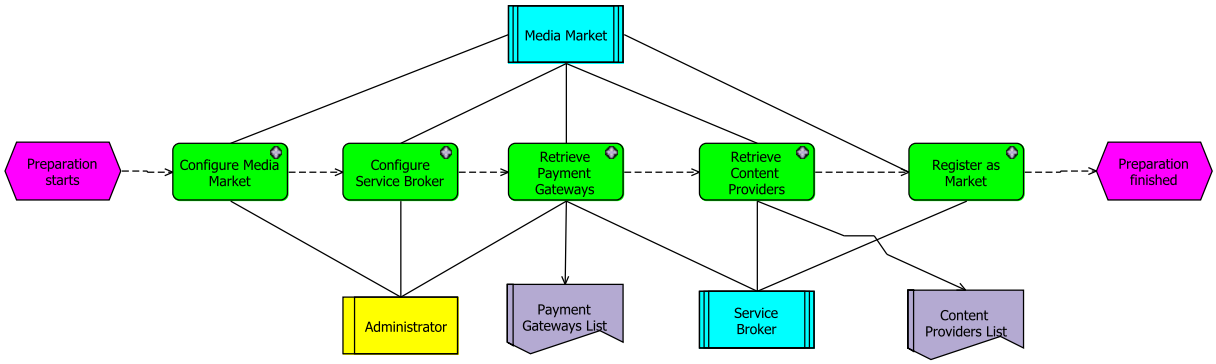


Figure F.11.: *Prepare*

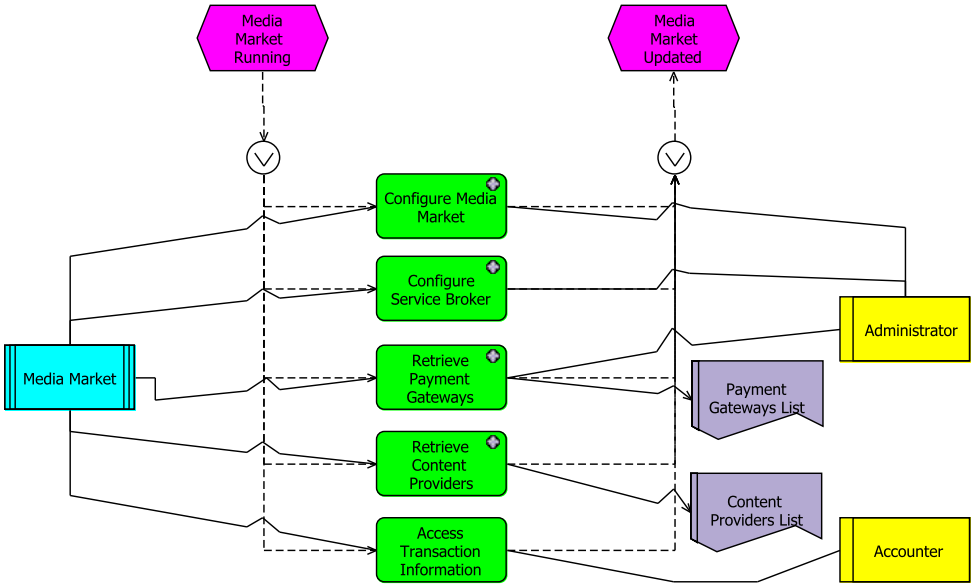


Figure F.12.: *System Access*

F.1.3.2. FAD

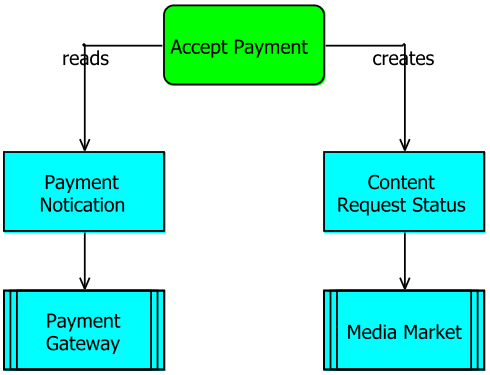


Figure F.13.: *FAD accept Payment*

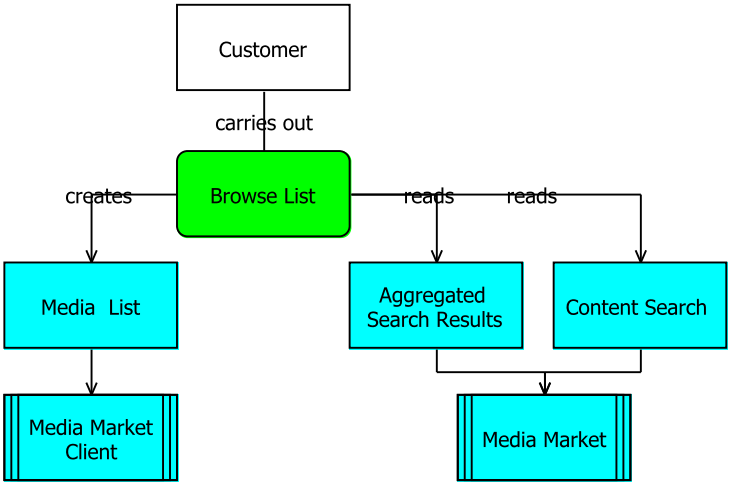


Figure F.14.: *Browse List*

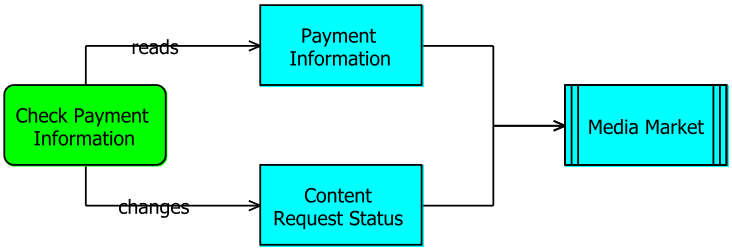


Figure F.15.: *Check Payment Information*

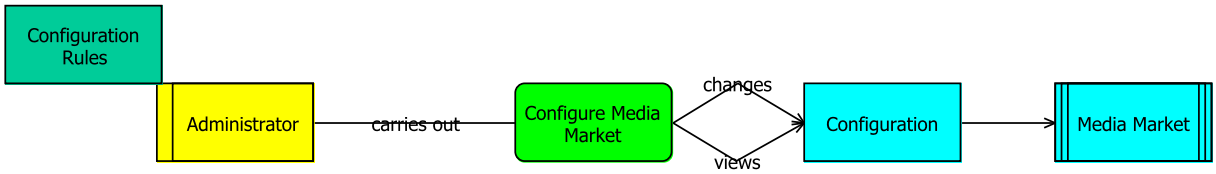


Figure F.16.: *Configure Media Market*

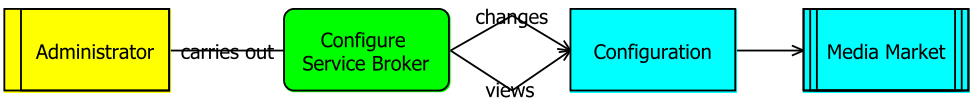


Figure F.17.: *Configure Service Broker*

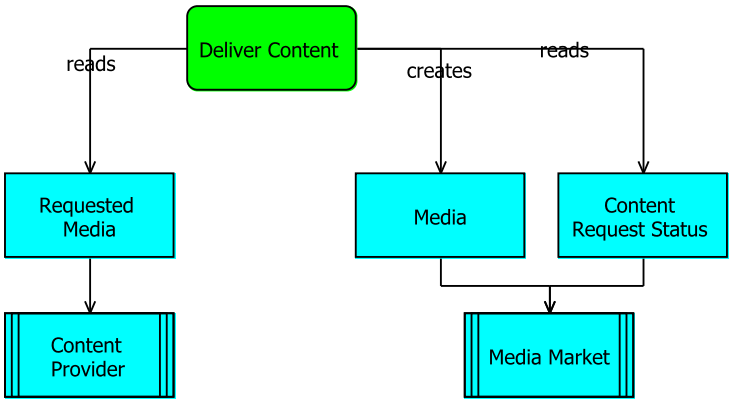


Figure F.18.: *Deliver Content*

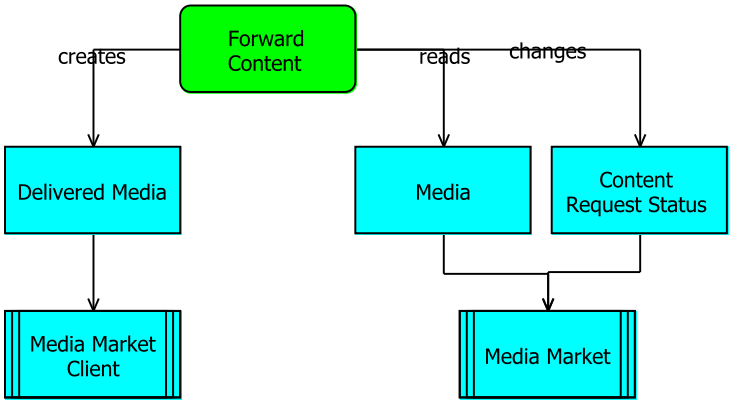


Figure F.19.: *Forward Content*

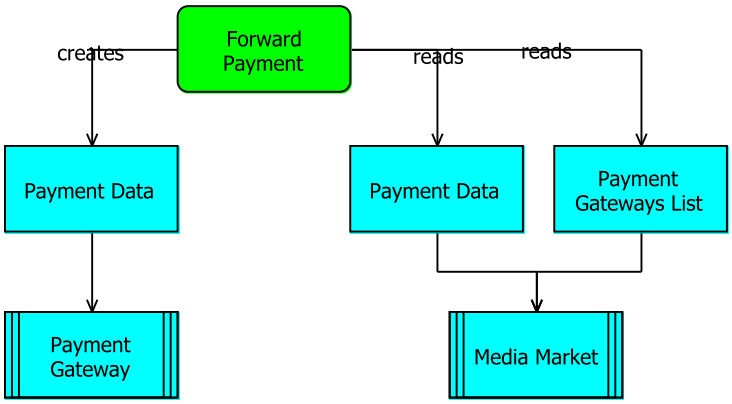


Figure F.20.: *ForwardPayment*

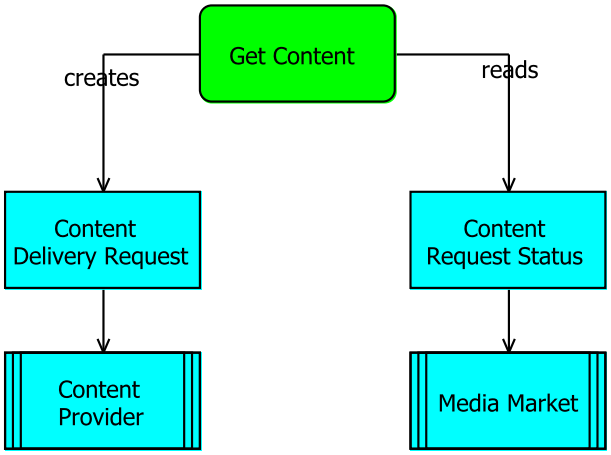


Figure F.21.: *Get Content*

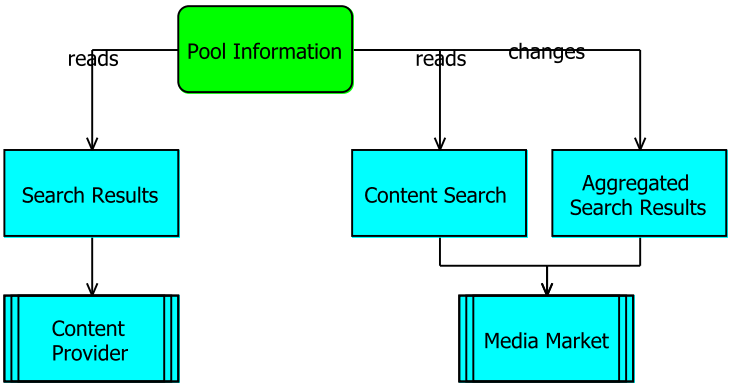


Figure F.22.: *Pool Information*

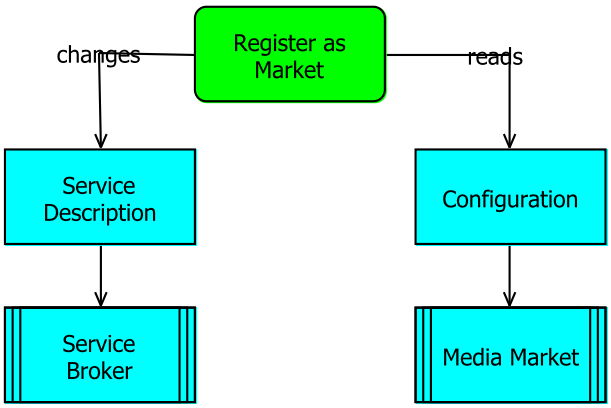


Figure F.23.: *Register As Market*

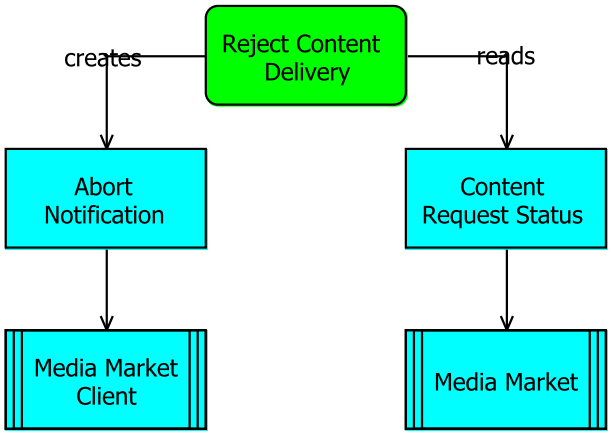


Figure F.24.: *Reject Content Delivery*

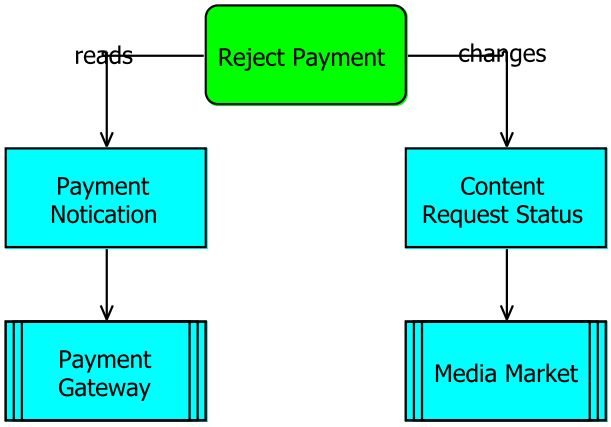


Figure F.25.: *Reject Payment*

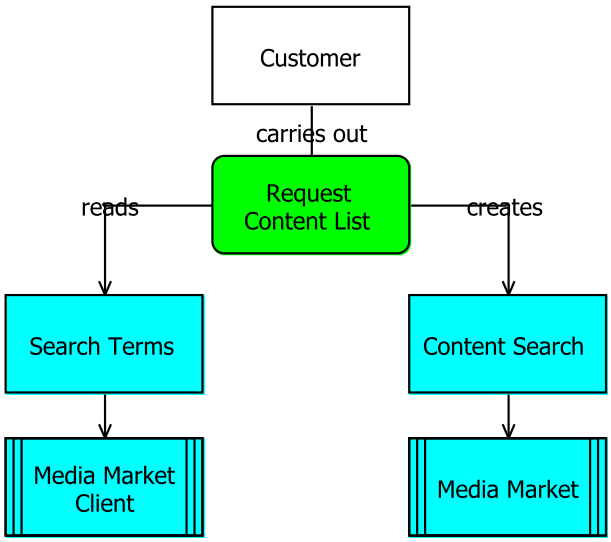


Figure F.26.: *Request Content List*

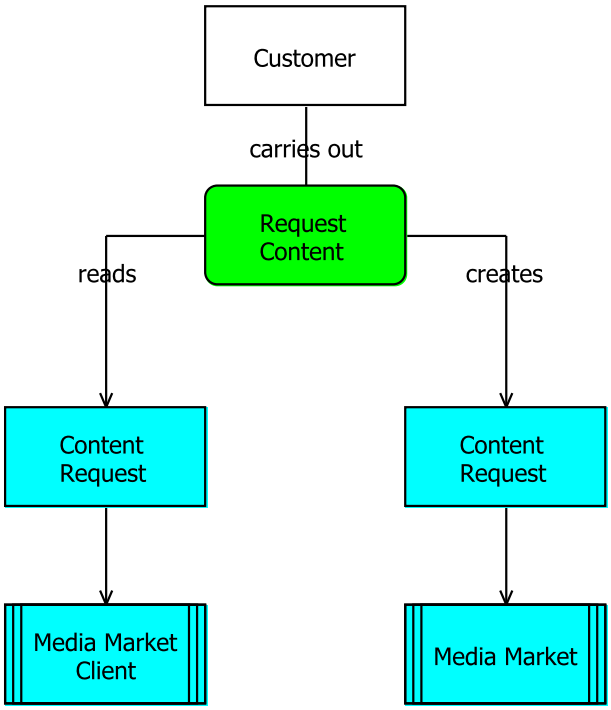


Figure F.27.: *Request Content*

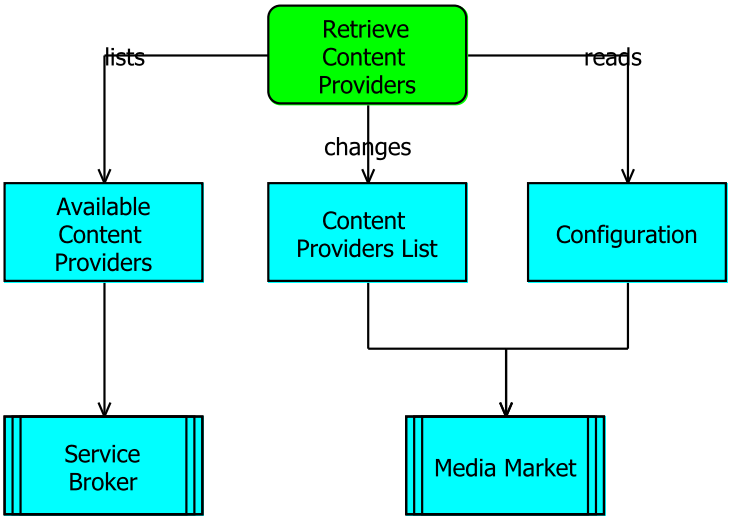


Figure F.28.: *Retrieve Content Providers*

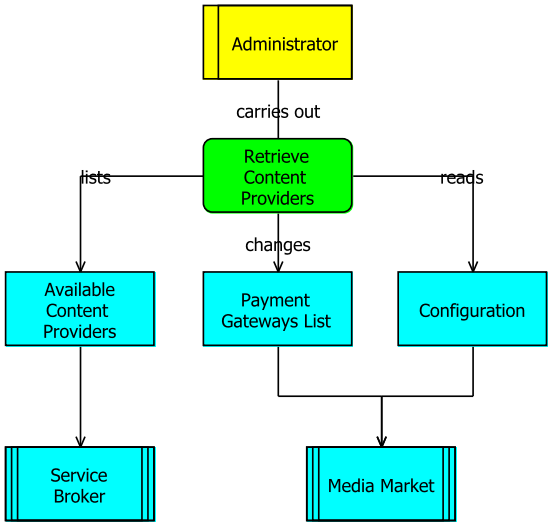


Figure F.29.: *Retrieve Payment Gateways*

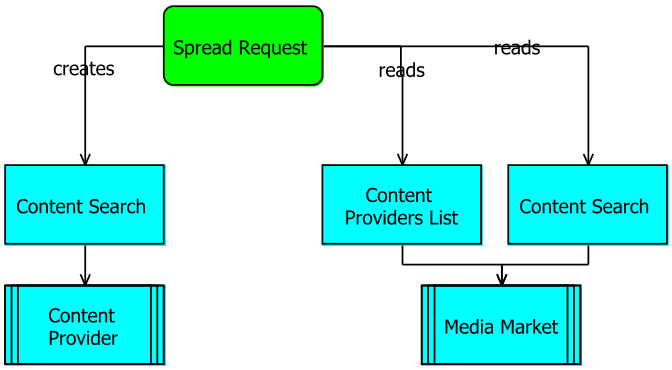


Figure F.30.: *Spread Request*

F.1.3.3. ERD

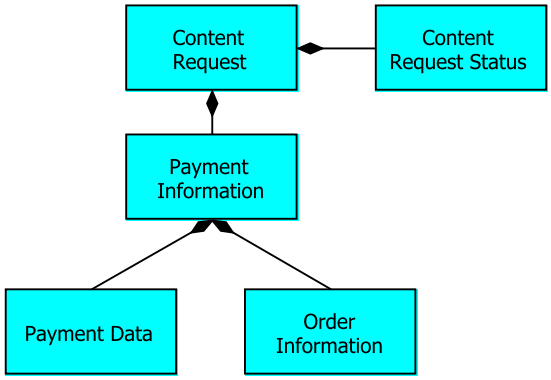


Figure F.31.: *Content Request*

F.1.4. Requirements

This appendix enumerates the textual requirements from the perspective of the customer (Appendix F.1.4.1), content provider (Appendix F.1.4.2[10]), and content aggregator (Appendix F.1.4.3[11]).

F.1.4.1. Customer

(R1[12]) **Request Content List** "The customer can request a content list."

(R2[13]) **Browse Content List** "The customer can browse the content list."

(R3[14]) **Request Content** "The customer can request content. He / She receives an abort notification or the content. The content is stored persistent."

(R4[15]) **Browse Owned Content** "The customer can browse his / her owned content."

(R5[16]) **View Owned Content** "The customer can view his / her owned content."

(R6[17]) **Search for Media Markets** "The customer can search for media markets using a service broker."

Security:

(R3A[18]) **Secure access flows via Internet (confidentiality)** "The access flows via the Internet must be secured in a way that the InternetAttacker is not able to threaten the confidentiality of the access flows."

(R3B[19]) **Secure access flows via Internet (integrity)** "The access flows via the Internet must be secured in a way that the InternetAttacker is not able to threaten the integrity of the access flows."

(R3C[20]) **Availability content request transmission** "The availability of the MMRequestContent machine and the transmission way must be 99.9%"

Compliance:

(R3D[21]) **Abroad Transfer** "Before actually contacting the Media Market, a consent has to be requested. The consent is displayed using the Media Market client. In case the Customer gives the consent, the payment data is checked later on."

(R3E[22]) **Data Minimization** "The content request must only contain private really necessary for processing the request."

[10] Page 529
[11] Page 529
[12] Page 1
[13] Page 1
[14] Page 1
[15] Page 1
[16] Page 1
[17] Page 1
[18] Page 1
[19] Page 1
[20] Page 1
[21] Page 1
[22] Page 1

(R3F[23]) **No 3rd Party Access** "It has to be assured that the private data contained in the content requests cannot be accessed by unauthorized third parties."

Costs:

(R3G[24]) **Running Costs** "The costs for receiving a request must be under 0.01 €."

(R3H[25]) **Maintenance Costs** "The costs for maintaining this the receiving of content requests must be under 2 person hours a month."

F.1.4.2. Content Provider

(R7[26]) **Receive Search Notification** "The content provider service can receive a notification that there is a new search request."

(R8[27]) **Answer Search Request** "The content provider service answers a search with a list of matching content."

(R9[28]) **Add Content** "The content provider can add new content to his / her list of content provided."

(R10[29]) **Modify Content** "The content provider can modify the content in his / her list of content provided."

(R11[30]) **Delete Content** "The content provider can delete content from his / her list of content provided."

(R12[31]) **Register Service** "The content provider can register his/her service at a service broker."

F.1.4.3. Content Aggregator

(R13[32]) **Spread Search Request** "The media market spreads content list requests from customer to registered content provider, which are obtained from a service broker."

(R14[33]) **Pool and Forward Answers** "The media market pools answers send by content provider to a content request and forwards the pooled information to the customer."

(R15[34]) **Check Payment Data** "The media market forwards payment data to payment gateways to check the validity of the payment data."

(R16[35]) **Reject Payment** "The media market stores a negative response to a payment request by the payment gateway in the content request status."

[23] Page 1
[24] Page 1
[25] Page 1
[26] Page 1
[27] Page 1
[28] Page 1
[29] Page 1
[30] Page 1
[31] Page 1
[32] Page 1
[33] Page 1
[34] Page 1
[35] Page 1

(R17[36]) **Accept Payment** "The media market stores a positive response to a payment request by the payment gateway in the content request status."

(R18[37]) **Check Payment Information** "The media market checks the payment information for obviously invalid information and updates the content request status accordingly."

(R20[38]) **Deliver Content** "The media market delivers content to the customer in case that there is a positive response from the payment gateway."

(R21[39]) **Search and Store Payment Gateways** "The content Aggregator can search for payment gateways using a service broker. The result is stored persistent."

(R22[40]) **Set Payment Gateway** "The content Aggregator can decide and set which payment gateway is to be used for validating payment data."

(R23[41]) **Save Content Requests** "The media market saves content requests."

(R24[42]) **Update Fulfillment Status for Content Requests** "The media market updates content requests according to their fulfillment status."

(R25[43]) **Access Content Requests** "The content Aggregator can access the content request, e.g. for billing purposes."

Compliance:

(R15A[44]) **Abroad Transfer** "Before actually contacting the payment gateway, a consent has to be requested. The consent is displayed using the Media Market client. In case the Customer gives the consent, the payment data is checked."

(R15B[45]) **Data Minimization** "The payment data send must only contain private really necessary for processing the request."

(R15C[46]) **No 3rd Party Access** "It has to be assured that the private data contained in the payment data cannot be accessed by unauthorized third parties."

Costs:

(R15D[47]) **Running Costs** "The costs for checking payment data must be under 0.01 €."

(R15E[48]) **Maintenance Costs** "The costs for maintaining the check payment data function must be under 2 person hours a month."

(R16A[49]) **Running Costs** "The costs for rejecting a payment must be under 0.001 €."

[36] Page 1
[37] Page 1
[38] Page 1
[39] Page 1
[40] Page 1
[41] Page 1
[42] Page 1
[43] Page 1
[44] Page 1
[45] Page 1
[46] Page 1
[47] Page 1
[48] Page 1
[49] Page 1

(R16B[50]) **Maintenance Costs** "The costs for rejecting a payment must be under 0.5 person hours a month."

(R17A[51]) **Running Costs** "The costs for accepting a payment must be under 0.001 €."

(R17B[52]) **Maintenance Costs** "The costs for accepting a payment must be under 0.5 person hours a month."

(R18A[53]) **Running Costs** "The costs for checking payment data must be under 0.01 €."

(R18B[54]) **Maintenance Costs** "The costs for maintaining the check payment data function must be under 2 person hours a month."

F.1.5. PresSuRE

This appendix contains the results of applying the PresSuRE steps connection domain discovery (Appendix F.1.5.1), security element elicitation (Appendix F.1.5.2[55]), and attacker elicitation (Appendix F.1.5.3[56]) to our running example.

F.1.5.1. Connection Domain Discovery

[50]Page 1
[51]Page 1
[52]Page 1
[53]Page 1
[54]Page 1
[55]Page 534
[56]Page 538

Table F.26.: *Connection Domain Discovery Table*

Domain 1	Domain 2	phenomena	Q1^{+}	Name?	Q2$^{\bullet}$	Name?	Q3°	Name?	PD*
MMCheckPayment-Information	PaymentInformation	MMCPI!{mmcpi-ReadForCheck-PaymentInformation} CRS!{Payment-Information}	No	-	No	-	No	-	R18 Check Payment - Information
MMCheckPayment-Information	ContentRequestStatus	MMCPI!{mmcpi-UpdateForCheck-PaymentInformation} CRS!{Content-RequestStatus}	No	-	No	-	No	-	R18 Check Payment - Information
MMPaymentData-Checker	PaymentGateway	MMPDC!{check-PaymentData}	Yes	VPN	No	-	No	-	R15 Forward - Payment
MMPaymentData-Checker	PaymentData	MMPDC!{get-PaymentData} P-D!{PaymentData}	No	-	No	-	No	-	R15 Forward - Payment
PaymentGateway	MMRejectPayment	PG!{pgReject-Payment}	Yes	VPN	No	-	No	-	R16 Reject Payment
PaymentGateway	MMAcceptPayment	PG!{pgAccept-Payment}	Yes	VPN	No	-	No	-	R17 Accept Payment
MMRejectPayment	ContentRequestStatus	MMRP!{mmrp-UpdateForReject-Payment} CRS!{-ContentRequest-Status}	No	-	No	-	No	-	R16 Reject Payment
ContentRequestStatus	MMAcceptPayment	MMAP!{mmrp-UpdateForAccept-Payment} CRS!{-ContentRequest-Status}	No	-	No	-	No	-	R17 Accept Payment
Customer	MediaMarketClient	C!{cSendContent-RequestForRequest-Content}	No	-	No	-	No	-	R3 Request Content
MediaMarketClient	MMRequestContent	MMC!{mmcForward-SendContentRequest-ForRequestContent}	Yes	Inter-net	No	-	No	-	R3 Request Content

Table F.26.: *Connection Domain Discovery Table*

Domain 1	Domain 2	phenomena	Q1[+]	Name?	Q2[•]	Name?	Q3[°]	Name?	PD[*]
ContentRequest	MMRequestContent	MMRC!{mmrcCreate-ForRequestContent} CR!{ContentRequest}	No	-	No	-	No	-	R3 Request Content

[+]Q1: Information transmitting domain involved? [•]Q2: Domain displaying information involved?
[°]Q3: Domain storing information involved? [*]PD: Related problem diagrams

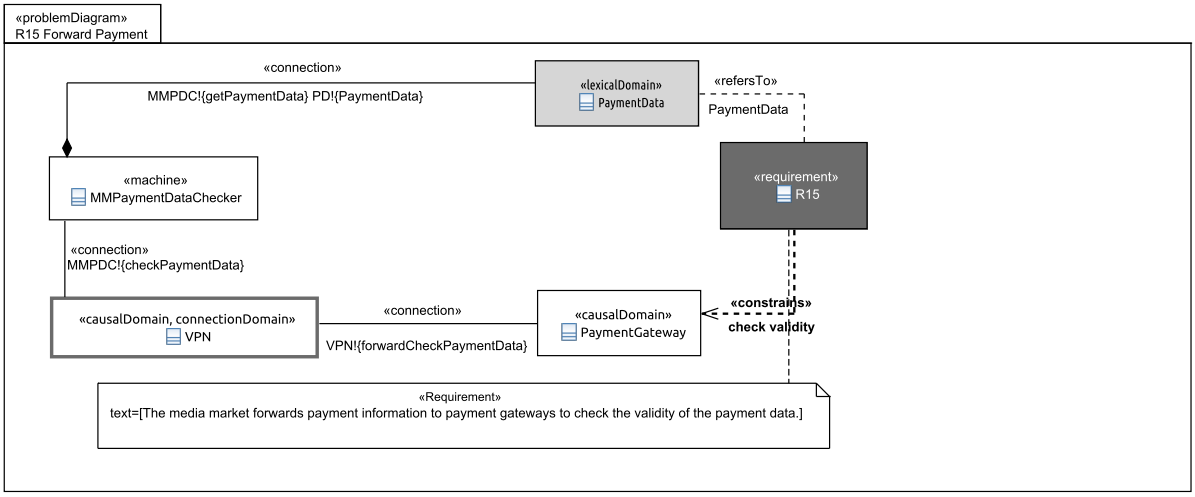


Figure F.32.: *Modified Problem Diagram for R15*

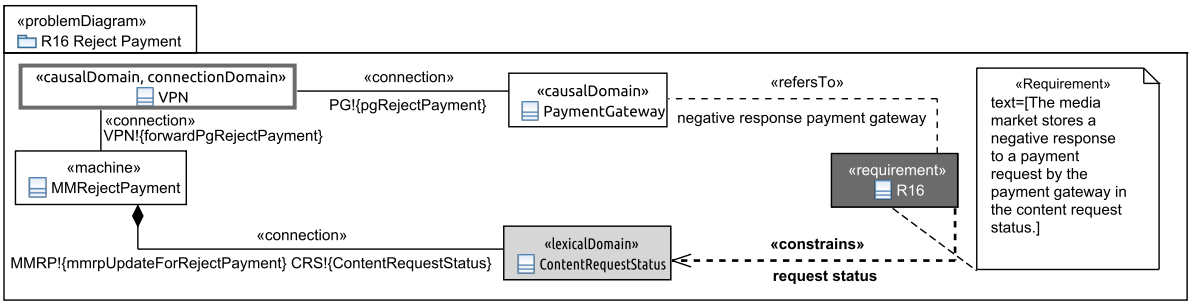


Figure F.33.: *Modified Problem Diagram for R16*

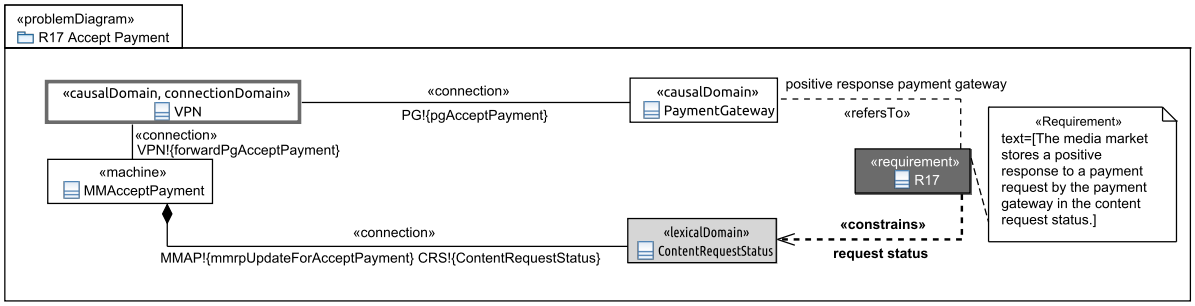


Figure F.34.: *Modified Problem Diagram for R17*

F.1.5.2. Security Element Elicitation

AssetCandidate	is asset	AuthorizedEntityCandidate	is authorized entity	rights
ContentRequest	☑	PaymentGateway	□	□ read □ write
		Internet	□	□ read □ write
		MediaMarketClient	☑	☒ read ☒ write
		Accounter	☑	☒ read □ write
		ContentProvider	□	□ read □ write
		Customer	☑	☒ read ☒ write
		Administrator	□	□ read □ write
		VPN	□	□ read □ write
		ServiceBroker	□	□ read □ write

☑ : yes, ☒ : must have, ⊡ : might have

Table F.27.: *Security Element Elicitation Table for ContentRequest*

AssetCandidate	is asset	StakeholderCandidate	is stakeholder	rights
ContentRequestStatus	☑	PaymentGateway	□	□ read □ write
		Internet	□	□ read □ write
		MediaMarketClient	□	□ read □ write
		Accounter	□	□ read □ write
		ContentProvider	□	□ read □ write
		Customer	☑	☒ read □ write
		Administrator	☑	☒ read □ write
		VPN	□	□ read □ write
		ServiceBroker	□	□ read □ write

☑ : yes, ☒ : must have, ⊡ : might have

Table F.28.: *Security Element Elicitation Table for ContentRequestStatus*

AssetCandidate	is asset	StakeholderCandidate	is stakeholder	rights
Internet	☐	PaymentGateway	☐	☐ know state ☐ control
		Internet	☐	☐ know state ☐ control
		MediaMarketClient	☐	☐ know state ☐ control
		Accounter	☐	☐ know state ☐ control
		ContentProvider	☐	☐ know state ☐ control
		Customer	☐	☐ know state ☐ control
		Administrator	☐	☐ know state ☐ control
		VPN	☐	☐ know state ☐ control
		ServiceBroker	☐	☐ know state ☐ control

☑ : yes, ☒ : must have, ⊡ : might have

Table F.29.: *Security Element Elicitation Table for Internet*

AssetCandidate	is asset	StakeholderCandidate	is stakeholder	rights
MediaMarketClient	☐	PaymentGateway	☐	☐ know state ☐ control
		Internet	☐	☐ know state ☐ control
		MediaMarketClient	☐	☐ know state ☐ control
		Accounter	☐	☐ know state ☐ control
		ContentProvider	☐	☐ know state ☐ control
		Customer	☐	☐ know state ☐ control
		Administrator	☐	☐ know state ☐ control
		VPN	☐	☐ know state ☐ control
		ServiceBroker	☐	☐ know state ☐ control

☑ : yes, ☒ : must have, ⊡ : might have

Table F.30.: *Security Element Elicitation Table for MediaMarketClient*

AssetCandidate	is asset	StakeholderCandidate	is stakeholder	rights
PaymentData	☑	PaymentGateway	☑	☒ read ☐ write
		Internet	☐	☐ read ☐ write
		MediaMarketClient	☑	☒ read ☒ write
		Accounter	☑	☒ read ☐ write
		ContentProvider	☐	☐ read ☐ write
		Customer	☑	☒ read ☒ write
		Administrator	☐	☐ read ☐ write
		VPN	☐	☐ read ☐ write
		ServiceBroker	☐	☐ read ☐ write

☑ : yes, ☒ : must have, ⊡ : might have

Table F.31.: *Security Element Elicitation Table for PaymentData*

AssetCandidate	is asset	StakeholderCandidate	is stakeholder	rights
PaymentGateway	☐	PaymentGateway	☐	☐ know state ☐ control
		Internet	☐	☐ know state ☐ control
		MediaMarketClient	☐	☐ know state ☐ control
		Accounter	☐	☐ know state ☐ control
		ContentProvider	☐	☐ know state ☐ control
		Customer	☐	☐ know state ☐ control
		Administrator	☐	☐ know state ☐ control
		VPN	☐	☐ know state ☐ control
		ServiceBroker	☐	☐ know state ☐ control

☑ : yes, ☒ : must have, ⊡ : might have

Table F.32.: *Security Element Elicitation Table for PaymentGateway*

AssetCandidate	is asset	StakeholderCandidate	is stakeholder	rights
PaymentInformation	☑	PaymentGateway	☑	☒ read ☐ write
		Internet	☐	☐ read ☐ write
		MediaMarketClient	☐	☐ read ☐ write
		Accounter	☑	☒ read ☐ write
		ContentProvider	☐	☐ read ☐ write
		Customer	☑	☒ read ☒ write
		Administrator	☐	☐ read ☐ write
		VPN	☐	☐ read ☐ write
		ServiceBroker	☐	☐ read ☐ write

☑ : yes, ☒ : must have, ⊡ : might have

Table F.33.: *Security Element Elicitation Table for PaymentInformation*

AssetCandidate	is asset	StakeholderCandidate	is stakeholder	rights
VPN	☐	PaymentGateway	☐	☐ know state ☐ control
		Internet	☐	☐ know state ☐ control
		MediaMarketClient	☐	☐ know state ☐ control
		Accounter	☐	☐ know state ☐ control
		ContentProvider	☐	☐ know state ☐ control
		Customer	☐	☐ know state ☐ control
		Administrator	☐	☐ know state ☐ control
		VPN	☐	☐ know state ☐ control
		ServiceBroker	☐	☐ know state ☐ control

☑ : yes, ☒ : must have, ⊡ : might have

Table F.34.: *Security Element Elicitation Table for VPN*

F.1.5.3. Attacker Elicitation

AttackCandidate	is attackable	Attacker	Name	Abilities
ContentRequest	☐	☐ network attacker		☐ read ☐ write ☐ interfere
		☐ physical attacker		☐ read ☐ write ☐ interfere
		☐ social attacker		☐ read ☐ write ☐ interfere
		☐ software attacker		☐ read ☐ write ☐ interfere

☑ : yes, ☒ : has the ability

Table F.35.: *Attacker Elicitation Table for ContentRequest*

AttackCandidate	is attackable	Attacker	Name	Abilities
ContentRequestStatus	☐	☐ network attacker		☐ read ☐ write ☐ interfere
		☐ physical attacker		☐ read ☐ write ☐ interfere
		☐ social attacker		☐ read ☐ write ☐ interfere
		☐ software attacker		☐ read ☐ write ☐ interfere

☑ : yes, ☒ : has the ability

Table F.36.: *Attacker Elicitation Table for ContentRequestStatus*

AttackCandidate	is attackable	Attacker	Name	Abilities
Customer	☐	☐ network attacker		☐ read ☐ write ☐ interfere
		☐ physical attacker		☐ read ☐ write ☐ interfere
		☐ social attacker		☐ read ☐ write ☐ interfere
		☐ software attacker		☐ read ☐ write ☐ interfere

☑ : yes, ☒ : has the ability

Table F.37.: *Attacker Elicitation Table for Customer*

AttackCandidate	is attackable	Attacker	Name	Abilities
Internet	☑	☑ network attacker	Internet Attacker	☐ read ☒ write ☒ interfere
		☐ physical attacker		☐ read ☐ write ☐ interfere
		☐ social attacker		☐ read ☐ write ☐ interfere
		☐ software attacker		☐ read ☐ write ☐ interfere

☑ : yes, ☒ : has the ability

Table F.38.: *Attacker Elicitation Table for Internet*

AttackCandidate	is attackable	Attacker	Name	Abilities
MediaMarketClient	☐	☐ network attacker		☐ read ☐ write ☐ interfere
		☐ physical attacker		☐ read ☐ write ☐ interfere
		☐ social attacker		☐ read ☐ write ☐ interfere
		☐ software attacker		☐ read ☐ write ☐ interfere

☑ : yes, ☒ : has the ability

Table F.39.: *Attacker Elicitation Table for MediaMarketClient*

AttackCandidate	is attackable	Attacker	Name	Abilities
PaymentData	☐	☐ network attacker		☐ read ☐ write ☐ interfere
		☐ physical attacker		☐ read ☐ write ☐ interfere
		☐ social attacker		☐ read ☐ write ☐ interfere
		☐ software attacker		☐ read ☐ write ☐ interfere

☑ : yes, ☒ : has the ability

Table F.40.: *Attacker Elicitation Table for PaymentData*

AttackCandidate	is attackable	Attacker	Name	Abilities
PaymentGateway	☐	☐ network attacker		☐ read ☐ write ☐ interfere
		☐ physical attacker		☐ read ☐ write ☐ interfere
		☐ social attacker		☐ read ☐ write ☐ interfere
		☐ software attacker		☐ read ☐ write ☐ interfere

☑ : yes, ☒ : has the ability

Table F.41.: *Attacker Elicitation Table for PaymentGateway*

AttackCandidate	is attackable	Attacker	Name	Abilities
PaymentInformation	☐	☐ network attacker		☐ read ☐ write ☐ interfere
		☐ physical attacker		☐ read ☐ write ☐ interfere
		☐ social attacker		☐ read ☐ write ☐ interfere
		☐ software attacker		☐ read ☐ write ☐ interfere

☑ : yes, ☒ : has the ability

Table F.42.: *Attacker Elicitation Table for PaymentInformation*

AttackCandidate	is attackable	Attacker	Name	Abilities
VPN	☐	☐ network attacker		☐ read ☐ write ☐ interfere
		☐ physical attacker		☐ read ☐ write ☐ interfere
		☐ social attacker		☐ read ☐ write ☐ interfere
		☐ software attacker		☐ read ☐ write ☐ interfere

☑ : yes, ☒ : has the ability

Table F.43.: *Attacker Elicitation Table for VPN*

F.1.6. Alternatives

Table F.44 shows the security relaxation template instance for SR3C.

Quality: Security, Requirement SR3C, Alternatives SR3O, SR3P, SR3Q	Property (CEM)	Possible Values	Value Original Requirement	Upper/Lower Bound	Value SR3O	Value SR3P	Value SR3Q
	Preparation time	one day, one week, two weeks, one month, two months, three months, four months, five months, six months, more than six months	four months	four months	four months	four months	four months
	Attack time	one day, one week, two weeks, one month, two months, three months, four months, five months, six months, more than six months	one day	one day	one day	one day	one day
	Specialist expertise	laymen, proficient, expert, multiple experts	experts	experts	experts	experts	experts
	Knowledge of the TOE	public, restricted, sensitive, critical	public	public	public	public	public
	Window of opportunity	unnecessary / unlimited, easy, moderate, difficult	easy	easy	easy	easy	easy
	IT hardware/software or other equipment	standard, specialized, bespoke, multiple bespoke	multiple bespoke	multiple bespoke	multiple bespoke	multiple bespoke	multiple bespoke
	Availability Specific						
	Degree of Availability	0% - 100%	99.99	98	99.5	99	98

Table F.44.: *Security Relaxation Template Instantiation for SR3C*

F.1.7. Valuation

Figure F.35 and Figure F.36 shows the content aggregator sub-network and payment gateway provider sub-network modeled in superdecisions.

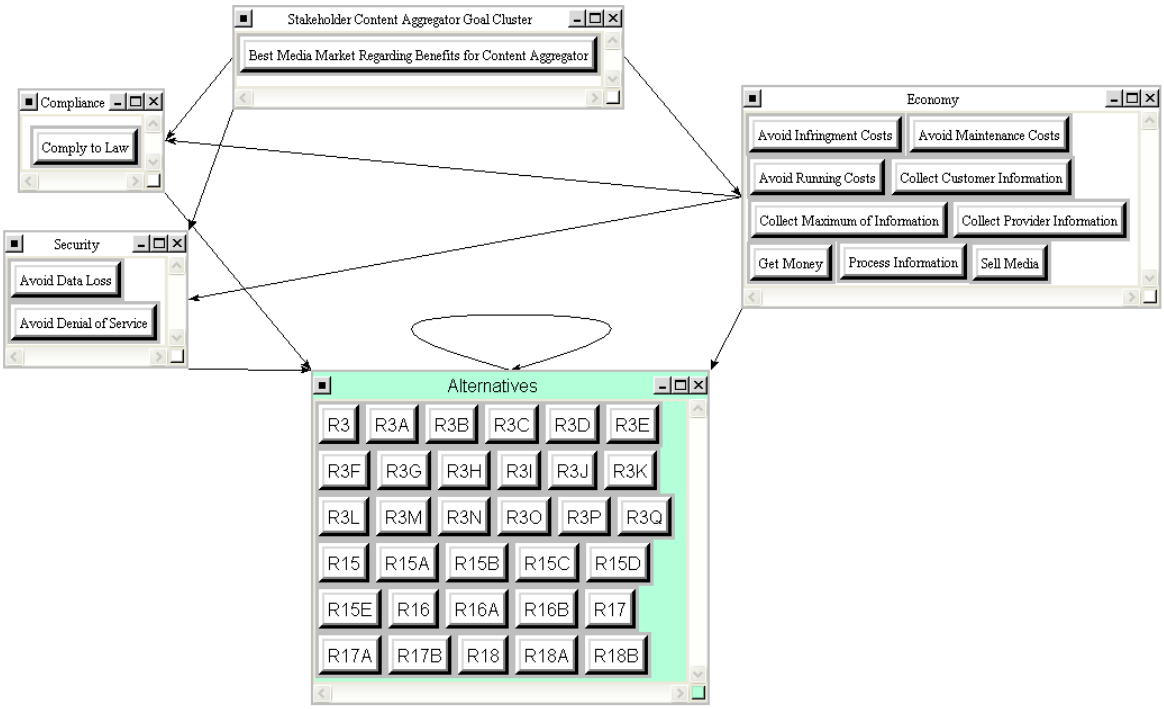


Figure F.35.: *Stakeholder Content Aggregator Sub-Network modeled in superdecisions*

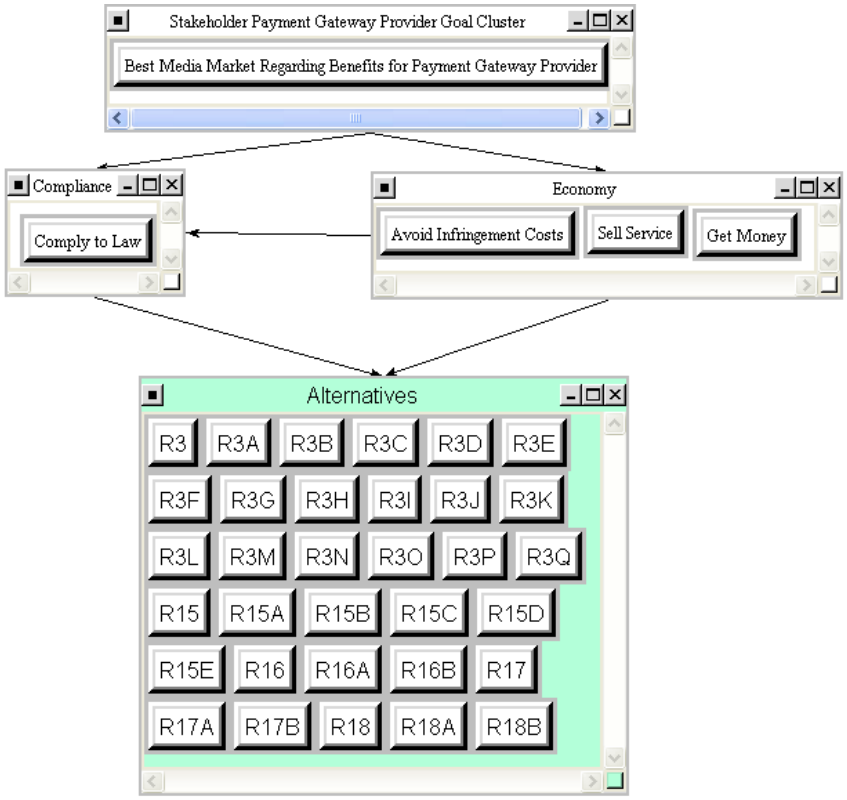


Figure F.36.: *Stakeholder Payment Gateway Provider Sub-Network modeled in superdecisions*

F.1.8. Optimization Model

Listing F.1: *Zimpl Model*

```
set G :=  { "economy_media_market",
            "make_profit_media_market",
            "sell_media_market",
            "get_money_media_market",
            "sell_media_media_market",
            "sell_info_media_market",
            "process_info_media_market",
            "collect_info_media_market",
            "collect_customer_info_media_market",
            "collect_provider_info_media_market",
            "collect_maximal_info_media_market",
            "avoid_costs_media_market",
            "avoid_running_costs_media_market",
            "avoid_management_media_market",
            "avoid_maintainance_costs_media_market",
            "avoid_infringment_costs_media_market",
            "security_media_market",
            "data_loss_media_market",
            "denial_of_service_media_market",
            "compliance_media_market",
            "law_media_market",
            "economy_customer",
            "security_customer",
            "privacy_customer",
            "fun_customer",
```

```
26                      "save_money",
27                      "low_bill",
28                      "additional_costs",
29                      "find_content_customer",
30                      "get_content_customer",
31                      "view_content_customer",
32                      "pay_bill",
33                      "manipulate_consumption",
34                      "manipulate_bill",
35                      "private_disclosed",
36                      "private_authorized",
37                      "private_other",
38                      "private_minimize",
39                      "private_protect",
40                      "private_protect_storage",
41                      "economy_payment_gateway_provider",
42                      "make_profit_payment_gateway_provider",
43                      "sell_payment_gateway_provider",
44                      "get_money_payment_gateway_provider",
45                      "sell_media_payment_gateway_provider",
46                      "avoid_costs_payment_gateway_provider",
47                      "avoid_management_payment_gateway_provider",
48                      "avoid_maintainance_costs_payment_gateway_provider",
49                      "avoid_infringment_costs_payment_gateway_provider",
50                      "security_payment_gateway_provider",
51                      "data_loss_payment_gateway_provider",
52                      "denial_of_service_payment_gateway_provider",
53                      "compliance_payment_gateway_provider",
54                      "law_payment_gateway_provider"};
55 set P := { "Content Payment" };
56 set A := { "Reject Payment", "Accept Payment", "Request Content", "Check Payment
       ↪ Information", "Forward Payment"};
57 set FR := { "R3", "R15", "R16", "R17", "R18"};
58 set QR := {"R3A", "R3B", "R3C", "R3D", "R3E", "R3F", "R3G", "R3H", "R3I", "R3J", "
       ↪ R3K", "R3L", "R3M", "R3N", "R3O", "R3P", "R3Q", "R15A", "R15B", "R15C", "
       ↪ R15D", "R15E", "R16A", "R16B", "R17A", "R17B", "R18A", "R18B"};
59 set R := FR + QR;
60
61 param gMust[G] := <"economy_media_market"> 0 default 0;
62 param pMust[P] := <"Content Payment"> 1 default 0;
63 param rMust[R] := <"R3"> 1, <"R15"> 1, <"R16"> 1, <"R17"> 1 default 0;
64
65 param gRoot[G] := <"economy_media_market"> 1, <"security_media_market"> 1, <"
       ↪ compliance_media_market"> 1, <"economy_customer"> 1, <"security_customer">
       ↪ 1, <"privacy_customer"> 1, <"fun_customer"> 1, <"
       ↪ economy_payment_gateway_provider"> 1, <"security_payment_gateway_provider">
       ↪ 1, <"compliance_payment_gateway_provider"> 1 default 0;
66
67 param rInitial[R] := <"R3"> 1, <"R15"> 1, <"R16"> 1, <"R17"> 1, <"R3B"> 1, <"R3C
       ↪ "> 1, <"R3D"> 1, <"R3E"> 1, <"R3F"> 1, <"R3G"> 1, <"R3H"> 1, <"R15A"> 1, <"
       ↪ R15B"> 1, <"R15C"> 1, <"R15D"> 1, <"R15E"> 1, < "R16A"> 1, <"R16B"> 1, <"
       ↪ R17A"> 1, <"R17B"> 1, <"R18A"> 1, <"R18B"> 1 default 0;
68 param rValue[R] := <"R3">0.3552,<"R3A">0.0907,<"R3B">0.631,<"R3C">0.7538,<"R3D
       ↪ ">0.2854,<"R3E">0.0684,<"R3F">0.0434,<"R3G">0.0128,<"R3H">0.0026,<"R3I
       ↪ ">0.0118,<"R3J">0.0079,<"R3K">0.0057,<"R3L">0.0074,<"R3M">0.0059,<"R3N
       ↪ ">0.0047,<"R3O">0.1679,<"R3P">0.0933,<"R3Q">0.0617,<"R15">0.116,<"R15A
       ↪ ">0.7178,<"R15B">0.0629,<"R15C">0.0412,<"R15D">0.0042,<"R15E">0.0005,<"R16
       ↪ ">0.0909,<"R16A">0.0046,<"R16B">0.0005,<"R17">0.1289,<"R17A">0.0046,<"R17B
       ↪ ">0.0005,<"R18">0.0343,<"R18A">0.0046,<"R18B">0.0005;
69
70 param g2gAND[G * G] := <"make_profit_media_market","sell_media_market"> 1,
71 <"make_profit_media_market","get_money_media_market"> 1,
```

```
72  <"make_profit_media_market","avoid_costs_media_market"> 1,
73  <"avoid_costs_media_market","avoid_management_media_market"> 1,
74  <"avoid_costs_media_market","avoid_maintainance_costs_media_market"> 1,
75  <"avoid_costs_media_market","avoid_infringment_costs_media_market"> 1,
76  <"avoid_costs_media_market","avoid_running_costs_media_market"> 1,
77  <"security_media_market","data_loss_media_market"> 1,
78  <"security_media_market","denial_of_service_media_market"> 1,
79  <"compliance_media_market","law_media_market"> 1,
80  <"save_money","low_bill"> 1,
81  <"save_money","additional_costs"> 1,
82  <"additional_costs","pay_bill"> 1,
83  <"security_customer","manipulate_consumption"> 1,
84  <"security_customer","manipulate_bill"> 1,
85  <"private_disclosed","private_authorized"> 1,
86  <"private_disclosed","private_other"> 1,
87  <"private_other","private_protect"> 1,
88  <"private_other","private_protect_storage"> 1,
89  <"fun_customer","find_content_customer"> 1,
90  <"fun_customer","get_content_customer"> 1,
91  <"fun_customer","view_content_customer"> 1,
92  <"make_profit_payment_gateway_provider","sell_payment_gateway_provider"> 1,
93  <"make_profit_payment_gateway_provider","get_money_payment_gateway_provider"> 1,
94  <"make_profit_payment_gateway_provider","avoid_costs_payment_gateway_provider"> 1,
95  <"avoid_costs_payment_gateway_provider","avoid_management_payment_gateway_provider
        ↪ "> 1,
96  <"avoid_costs_payment_gateway_provider","
        ↪ avoid_maintainance_costs_payment_gateway_provider"> 1,
97  <"avoid_costs_payment_gateway_provider","
        ↪ avoid_infringment_costs_payment_gateway_provider"> 1,
98  <"security_payment_gateway_provider","data_loss_payment_gateway_provider"> 1,
99  <"security_payment_gateway_provider","denial_of_service_payment_gateway_provider">
        ↪  1,
100 <"compliance_payment_gateway_provider","law_payment_gateway_provider"> 1
101 default 0;
102 param g2gXOR[G * G] := <"sell_media_market","make_profit_media_market"> 0 default
        ↪ 0;
103
104 param g2gOR[G * G] := <"economy_media_market", "make_profit_media_market"> 1,
105 <"sell_media_market", "sell_media_media_market"> 1,
106 <"sell_media_market", "sell_info_media_market"> 1,
107 <"sell_info_media_market", "process_info_media_market"> 1,
108 <"sell_info_media_market", "collect_info_media_market"> 1,
109 <"collect_info_media_market", "collect_maximal_info_media_market"> 1,
110 <"collect_info_media_market", "collect_customer_info_media_market"> 1,
111 <"collect_info_media_market", "collect_provider_info_media_market"> 1,
112 <"economy_customer","save_money"> 1,
113 <"privacy_customer", "private_disclosed"> 1,
114 <"private_authorized", "private_minimize"> 1,
115 <"economy_payment_gateway_provider", "make_profit_payment_gateway_provider"> 1,
116 <"sell_payment_gateway_provider", "sell_media_payment_gateway_provider"> 1
117 default 0;
118
119 param g2gDeny[G * G] := <"collect_customer_info_media_market","privacy_customer">
        ↪ 1 default 0;
120
121 param g2rAND[G * R] := <"law_media_market", "R3F"> 1,
122 <"law_media_market", "R15C">1 ,
123 <"get_money_media_market", "R15"> 1,
124 <"get_money_media_market", "R16"> 1,
125 <"get_money_media_market", "R17"> 1,
126 <"sell_media_media_market", "R3"> 1,
127 <"sell_media_media_market", "R17"> 1,
```

```
128 <"process_info_media_market", "R3"> 1,
129 <"process_info_media_market", "R18"> 1,
130 <"collect_provider_info_media_market", "R3"> 1,
131 <"collect_provider_info_media_market", "R18"> 1,
132 <"collect_customer_info_media_market", "R3"> 1,
133 <"get_content_customer","R3"> 1,
134 <"get_content_customer","R17"> 1
135 default 0;
136 param g2rXOR[G * R] := <"law_media_market", "R3D"> 1,
137 <"law_media_market", "R15A"> 1,
138 <"denial_of_service_media_market", "R3C"> 1,
139 <"denial_of_service_media_market", "R3O"> 1,
140 <"denial_of_service_media_market", "R3P"> 1,
141 <"denial_of_service_media_market", "R3Q"> 1,
142 <"avoid_running_costs_media_market","R3G"> 1,
143 <"avoid_running_costs_media_market","R3L"> 1,
144 <"avoid_running_costs_media_market","R3M"> 1,
145 <"avoid_running_costs_media_market","R3N"> 1,
146 <"private_protect","R3A"> 1,
147 <"private_protect","R3I"> 1,
148 <"private_protect","R3J"> 1,
149 <"private_protect","R3K"> 1
150 default 0;
151
152 param g2rOR[G * R] := <"law_media_market", "R15B"> 1,
153 <"law_media_market", "R3E"> 1,
154 <"denial_of_service_media_market", "R3B"> 1,
155 <"data_loss_media_market", "R3A"> 1,
156 <"data_loss_media_market", "R3I"> 1,
157 <"data_loss_media_market", "R3J"> 1,
158 <"data_loss_media_market", "R3K"> 1,
159 <"avoid_running_costs_media_market","R15D"> 1,
160 <"avoid_running_costs_media_market","R16A"> 1,
161 <"avoid_running_costs_media_market","R17A"> 1,
162 <"avoid_running_costs_media_market","R18A"> 1,
163 <"avoid_infringment_costs_media_market", "R18"> 1,
164 <"avoid_infringment_costs_media_market", "R3A"> 1,
165 <"avoid_infringment_costs_media_market", "R3B"> 1,
166 <"avoid_infringment_costs_media_market", "R3C"> 1,
167 <"avoid_infringment_costs_media_market", "R3D"> 1,
168 <"avoid_infringment_costs_media_market", "R3E"> 1,
169 <"avoid_infringment_costs_media_market", "R3F"> 1,
170 <"avoid_infringment_costs_media_market", "R3I"> 1,
171 <"avoid_infringment_costs_media_market", "R3J"> 1,
172 <"avoid_infringment_costs_media_market", "R3K"> 1,
173 <"avoid_infringment_costs_media_market", "R3O"> 1,
174 <"avoid_infringment_costs_media_market", "R3P"> 1,
175 <"avoid_infringment_costs_media_market", "R3Q"> 1,
176 <"avoid_infringment_costs_media_market", "R15A"> 1,
177 <"avoid_infringment_costs_media_market", "R15B"> 1,
178 <"avoid_infringment_costs_media_market", "R15C"> 1,
179 <"avoid_maintainance_costs_media_market", "R3B"> 1,
180 <"avoid_maintainance_costs_media_market", "R3C"> 1,
181 <"avoid_maintainance_costs_media_market", "R3O"> 1,
182 <"avoid_maintainance_costs_media_market", "R3P"> 1,
183 <"avoid_maintainance_costs_media_market", "R3Q"> 1,
184 <"avoid_maintainance_costs_media_market", "R3H"> 1,
185 <"avoid_maintainance_costs_media_market", "R15E"> 1,
186 <"avoid_maintainance_costs_media_market", "R16B"> 1,
187 <"avoid_maintainance_costs_media_market", "R17B"> 1,
188 <"avoid_maintainance_costs_media_market", "R18B"> 1,
189 <"get_money_media_market", "R3C"> 1,
```

```
<"get_money_media_market", "R3O"> 1,
<"get_money_media_market", "R3P"> 1,
<"get_money_media_market", "R3Q"> 1,
<"sell_media_media_market", "R3B"> 1,
<"sell_media_media_market", "R3C"> 1,
<"sell_media_media_market", "R3O"> 1,
<"sell_media_media_market", "R3P"> 1,
<"sell_media_media_market", "R3Q"> 1,
<"process_info_media_market", "R3C"> 1,
<"process_info_media_market", "R3O"> 1,
<"process_info_media_market", "R3P"> 1,
<"process_info_media_market", "R3Q"> 1,
<"collect_provider_info_media_market", "R3C"> 1,
<"collect_provider_info_media_market", "R3O"> 1,
<"collect_provider_info_media_market", "R3P"> 1,
<"collect_provider_info_media_market", "R3Q"> 1,
<"collect_maximal_info_media_market", "R3"> 1,
<"collect_maximal_info_media_market", "R16"> 1,
<"collect_maximal_info_media_market", "R17"> 1,
<"collect_maximal_info_media_market", "R18"> 1,
<"collect_maximal_info_media_market", "R3C"> 1,
<"collect_maximal_info_media_market", "R3O"> 1,
<"collect_maximal_info_media_market", "R3P"> 1,
<"collect_maximal_info_media_market", "R3Q"> 1,
<"collect_customer_info_media_market", "R16"> 1,
<"collect_customer_info_media_market", "R17"> 1,
<"collect_customer_info_media_market", "R18"> 1,
<"collect_customer_info_media_market", "R3C"> 1,
<"collect_customer_info_media_market", "R3O"> 1,
<"collect_customer_info_media_market", "R3P"> 1,
<"collect_customer_info_media_market", "R3Q"> 1,
<"pay_bill","R3B"> 1,
<"find_content_customer","R3C"> 1,
<"find_content_customer","R3O"> 1,
<"find_content_customer","R3P"> 1,
<"find_content_customer","R3Q"> 1,
<"get_content_customer","R3C"> 1,
<"get_content_customer","R3O"> 1,
<"get_content_customer","R3P"> 1,
<"get_content_customer","R3Q"> 1,
<"manipulate_consumption","R3B"> 1,
<"manipulate_bill","R3B"> 1,
<"private_minimize","R3E"> 1,
<"private_minimize","R15B"> 1,
<"private_protect_storage","R3D"> 1,
<"private_protect_storage","R15A"> 1,
<"private_protect_storage","R3F"> 1,
<"private_protect_storage","R15C"> 1,
<"private_protect","R15B"> 1
default 0;

param p2a[P * A] := <"Content Payment","Reject Payment"> 1, <"Content Payment","
    ↪ Accept Payment"> 1,  <"Content Payment","Request Content"> 1, <"Content
    ↪ Payment","Check Payment Information"> 1, <"Content Payment","Forward
    ↪ Payment"> 1 default 0;

param a2r[A * R] := <"Reject Payment", "R16"> 1, <"Accept Payment","R17"> 1, <"
    ↪ Request Content", "R3"> 1, <"Check Payment Information","R18"> 1, <"Forward
    ↪  Payment", "R15"> 1 default 0;

param r2rDeny[R * R] := <"R3A", "R3G"> 1,
<"R3A", "R3L"> 1,
```

```
247 <"R3A", "R3M"> 1,
248 <"R3A", "R3N"> 1,
249 <"R3I", "R3G"> 1,
250 <"R3I", "R3L"> 1,
251 <"R3K", "R3G"> 1,
252 <"R3C", "R3G"> 1,
253 <"R3C", "R3L"> 1,
254 <"R3C", "R3M"> 1,
255 <"R3C", "R3N"> 1,
256 <"R3O", "R3G"> 1
257 default 0;
258
259 param r2rRequire[R * R] := <"R3","R15"> 1,
260 <"R3", "R18"> 1,
261 <"R15", "R16"> 1,
262 <"R15", "R17"> 1
263 default 0;
264
265 param r2rComplements[FR * QR] := <"R3","R3A"> 1,
266 <"R3", "R3B"> 1,
267 <"R3", "R3C"> 1,
268 <"R3", "R3D"> 1,
269 <"R3", "R3E"> 1,
270 <"R3", "R3F"> 1,
271 <"R3", "R3G"> 1,
272 <"R3", "R3H"> 1,
273 <"R3", "R3I"> 1,
274 <"R3", "R3J"> 1,
275 <"R3", "R3K"> 1,
276 <"R3", "R3L"> 1,
277 <"R3", "R3M"> 1,
278 <"R3", "R3N"> 1,
279 <"R3", "R3O"> 1,
280 <"R3", "R3P"> 1,
281 <"R3", "R3Q"> 1,
282 <"R15", "R15A"> 1,
283 <"R15", "R15B"> 1,
284 <"R15", "R15C"> 1,
285 <"R15", "R15D"> 1,
286 <"R15", "R15E"> 1,
287 <"R16","R16A"> 1,
288 <"R16", "R16B"> 1,
289 <"R17", "R17A"> 1,
290 <"R17", "R17B"> 1,
291 <"R18", "R18A"> 1,
292 <"R18", "R18B"> 1
293 default 0;
294
295 param r2rAlternative[R * R] := <"R3D", "R15A"> 1,
296 <"R3A", "R3I"> 1,
297 <"R3A", "R3J"> 1,
298 <"R3A", "R3K"> 1,
299 <"R3I", "R3J"> 1,
300 <"R3I", "R3K"> 1,
301 <"R3K", "R3K"> 1,
302 <"R3C", "R3O"> 1,
303 <"R3C", "R3P"> 1,
304 <"R3C", "R3Q"> 1,
305 <"R3O", "R3P"> 1,
306 <"R3O", "R3Q"> 1,
307 <"R3P", "R3Q"> 1,
308 <"R3G", "R3L"> 1,
```

```
309 <"R3G", "R3M"> 1,
310 <"R3G", "R3N"> 1,
311 <"R3G", "R3L"> 1,
312 <"R3L", "R3M"> 1,
313 <"R3L", "R3N"> 1,
314 <"R3M", "R3N"> 1
315 default 0;
316
317 var g[G] binary;
318 var p[P] binary;
319 var a[A] binary;
320 var r[R] binary;
321
322
323 minimize dif: ((sum <k> in R : rInitial[k] * rValue[k]) - (sum <k> in R : r[k] *
    ↪ rValue[k])) - (sum <i> in G : g[i] * 0.000000001);
324
325 subto g2gParent: forall <i> in G: (1 - gRoot[i]) * (g[i] - (sum <j> in G: (g[j] *
    ↪ (g2gAND[j,i]+g2gOR[j,i]+g2gXOR[j,i])))) <= 0;
326
327 subto gMust:  forall <i> in G:  gMust[i] - g[i] <= 0;
328
329 subto g2gDeny: forall <i,j> in G cross G: (g2gDeny[i,j]  * (g[i] + g[j])) - 1 <=
    ↪ 0;
330
331 subto g2gAnd: forall<i,j> in G cross G: (g2gAND[i,j]  * (g[i] - g[j])) <= 0;
332
333 subto g2gOr:  forall<i> in G: (1-prod <j> in G : (1- g2gOR[i,j]))*((sum <j> in G :
    ↪   g2gOR[i,j] * g[j]) - g[i]) >= 0;
334
335 subto g2gXOr:  forall<i> in G: (1-prod <j> in G : (1- g2gXOR[i,j]))*((sum <j> in G
    ↪  :  g2gXOR[i,j] * g[j]) - g[i]) == 0;
336
337 subto pMust:  forall <m> in P:  pMust[m] - p[m] <= 0;
338
339 subto p2a: forall<m,n> in P cross A: (p2a[m,n]  * (p[m] - a[n])) <= 0;
340
341 subto a2r: forall<n,k> in A cross R: (a2r[n,k]  * (a[n] - r[k])) <= 0;
342
343 subto g2rAnd: forall<i,k> in G cross R: (g2rAND[i,k]  * (g[i] - r[k])) <= 0;
344
345 subto g2rOr:  forall<i> in G: (1-prod <k> in R : (1- g2rOR[i,k]))*((sum <k> in R :
    ↪   g2rOR[i,k] * r[k]) - g[i]) >= 0;
346
347 subto g2rXOr:  forall<i> in G: (1-prod <k> in R : (1- g2rXOR[i,k]))*((sum <k> in R
    ↪  :  g2rXOR[i,k] * r[k]) - g[i]) == 0;
348
349 subto rMust:  forall <k> in R:  rMust[k] - r[k] <= 0;
350
351 subto r2rDeny: forall <k,l> in R cross R: (r2rDeny[k,l]  * (r[k] + r[l])) - 1 <=
    ↪ 0;
352
353 subto r2rRequire: forall<k,l> in R cross R:  (r2rRequire[k,l]  * (r[l] - r[k])) <=
    ↪  0;
354
355 subto r2rComplements: forall<k,l> in FR cross QR: (r2rComplements[k,l]  * (r[l] -
    ↪ r[k])) <= 0;
356
357 subto r2rAlternative: forall <k,l> in R cross R: (r2rAlternative[k,l]  * (r[k] + r
    ↪ [l])) - 1 <= 0;
358
```

```
359 subto r2g: forall <k> in R: r[k] - (sum <i> in G: (g[i] * (g2rAND[i,k]+g2rOR[i,k]+
    ↪ g2rXOR[i,k]))) <= 0;
```

F.2. Smart Grid

This Case is part of the electronic version of this work due to printing restrictions.

F.3. Voting System

This Case is part of the electronic version of this work due to printing restrictions.

Bibliography

[1] Wil M. P. Van Der Aalst. Process-Aware Information Systems: Lessons to Be Learned from Process Mining. In Kurt Jensen and Wil M Aalst, editors, *Transactions on Petri Nets and Other Models of Concurrency II*, chapter Process-Aw, pages 1–26. Springer-Verlag, Berlin, Heidelberg, 2009. ISBN 978-3-642-00898-6. doi: 10.1007/978-3-642-00899-3_1. URL `http://dx.doi.org/10.1007/978-3-642-00899-3_1`.

[2] Jarmo J. Ahonen and Paula Savolainen. Software engineering projects may fail before they are started: Post-mortem analysis of five cancelled projects. *Journal of Systems and Software*, 83(11):2175 – 2187, 2010. ISSN 0164-1212. doi: 10.1016/j.jss.2010.06.023. URL `http://www.sciencedirect.com/science/article/pii/S0164121210001706`. Interplay between Usability Evaluation and Software Development.

[3] J. M. Van Den Akker, S. Brinkkemper, G. Diepen, and J. Versendaal. Determination of the next release of a software product: an approach using integer linear programming. In *Proceeding of the 11th International Workshop on Requirements Engineering: Foundation for Software Quality REFSQ'05*, pages 247–262, 2005.

[4] A. Alebrahim, D. Hatebur, and M. Heisel. Towards systematic integration of quality requirements into software architecture. In *Proc. ECSA'11*, pages 17–25. Springer, 2011.

[5] Azadeh Alebrahim, Denis Hatebur, and Maritta Heisel. A method to derive software architectures from quality requirements. In Tran Dan Thu and Karl Leung, editors, *Proceedings of the 18th Asia-Pacific Software Engineering Conference (APSEC)*, pages 322–330. IEEE Computer Society, 2011.

[6] Azadeh Alebrahim, Christine Choppy, Stephan Faßbender, and Maritta Heisel. Optimizing functional and quality requirements according to stakeholders' goals. In Ivan MistrikRami BahsoonPeter EelesRoshanak RoshandelMichael Stal, editor, *Relating System Quality and Software Architecture*, pages 75 – 120. Morgan Kaufmann, Boston, 2014. ISBN 978-0-12-417009-4. doi: 10.1016/b978-0-12-417009-4.00004-1. URL `http://www.sciencedirect.com/science/article/pii/B9780124170094000041`.

[7] Azadeh Alebrahim, Stephan Faßbender, Martin Filipczyk, Michael Goedicke, Maritta Heisel, and Marco Konersmann. Towards a computer-aided problem-oriented variability requirements engineering method. In *Advanced Information Systems Engineering Workshops - CAiSE 2014 International Workshops, Thessaloniki, Greece, June 16-20, 2014. Proceedings*, volume 178 of *Lecture Notes in Business Information Processing*, pages 136–147. Springer, 2014. ISBN 978-3-319-07868-7. doi: 10.1007/978-3-319-07869-4_12. URL `http://link.springer.com/chapter/10.1007/978-3-319-07869-4_12`.

[8] Azadeh Alebrahim, Stephan Faßbender, Maritta Heisel, and Rene Meis. Problem-based requirements interaction analysis. In *Requirements Engineering: Foundation for Software Quality 20th International Working Conference (REFSQ)*, volume 8396 of *Lecture Notes in Computer Science*, pages 200–215. Springer, 2014. ISBN 978-3-319-05842-9. doi: 10.1007/978-3-319-05843-6_15. URL `http://link.springer.com/chapter/10.1007%2F978-3-319-05843-6_15`.

[9] Azadeh Alebrahim, Stephan Faßbender, Martin Filipczyk, Michael Goedicke, and Maritta Heisel. Towards systematic selection of architectural patterns with respect to quality requirements. In *Proceedings of the 20th European Conference on Pattern Languages of Program*, EuroPLoP '15, pages –, New York, NY, USA, 2015. ACM. To be Published.

[10] Azadeh Alebrahim, Stephan Faßbender, Martin Filipczyk, Michael Goedicke, and Maritta Heisel. Relating performance and security tactics to architectural patterns. In *Proceedings of the 20th European Conference on Pattern Languages of Program*, EuroPLoP '15, pages –, New York, NY, USA, 2015. ACM. To be Published.

[11] Azadeh Alebrahim, Denis Hatebur, Stephan Faßbender, Ludger Goeke, and Isabelle Côté. A pattern-based and tool-supported risk analysis method compliant to iso 27001 for cloud systems. *Int. J. Secur. Softw. Eng.*, 6(1):24–46, January 2015. ISSN 1947-3036. doi: 10.4018/ijsse.2015010102.

[12] Christopher Alexander. *A Pattern Language: Towns, Buildings, Construction.* Oxford University Press, 1977.

[13] Ian Alexander and S. Robertson. Understanding project sociology by modeling stakeholders. *Software, IEEE*, 21(1):23–27, Jan 2004. ISSN 0740-7459. doi: 10.1109/ms.2004.1259199.

[14] D. Alrajeh, J. Kramer, A. Russo, and S. Uchitel. Learning operational requirements from goal models. In *ICSE '09*, pages 265–275, 2009.

[15] Young Alva, Aaron. L-square: Preliminary extension of the square methodology to address legal compliance. In *ESPRE'14 Workshop at RE'14*, 2014. doi: 10.1109/espre.2014.6890524.

[16] Daniel Amyot, Sepideh Ghanavati, Jennifer Horkoff, Gunter Mussbacher, Liam Peyton, and Eric Yu. Evaluating goal models within the goal-oriented requirement language. *International Journal of Intelligent Systems*, 25(8):841–877, 2010. ISSN 1098-111X. doi: 10.1002/int.20433. URL `http://dx.doi.org/10.1002/int.20433`.

[17] K. M. Annervaz, V. Kaulgud, S. Sengupta, and M. Savagaonkar. Natural language requirements quality analysis based on business domain models. In *Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on*, pages 676–681, Nov 2013. doi: 10.1109/ase.2013.6693132.

[18] James G. Apple and Robert P. Deyling. *A primer on the civil-law system [microform] / by James G. Apple and Robert P. Deyling.* Federal Judicial Center, [Washington, D.C.] :, 1995. URL `http://purl.access.gpo.gov/GPO/LPS55055`.

[19] A. Arsanjani, Liang-Jie Zhang, Michael Ellis, Abdul Allam, and Kishore Channabasavaiah. Design an SOA solution using a reference architecture. Technical report, IBM, 2007.

[20] A. Arsanjani, S. Ghosh, A. Allam, T. Abdollah, S. Gariapathy, and K. Holley. Soma: a method for developing service-oriented solutions. *IBM Systems Journal*, 47(3), 2008.

[21] Colin Atkinson. Supporting and applying the uml conceptual framework. In *The Unified Modeling Language.UML 98: Beyond the Notation*, volume 1618 of *Lecture Notes in Computer Science*, pages 21–36. Springer Berlin Heidelberg, 1999. ISBN 978-3-540-66252-5. doi: 10.1007/978-3-540-48480-6_3.

[22] Aybüke Aurum and Claes Wohlin. The fundamental nature of requirements engineering activities as a decision-making process. *Information and Software Technology*, 45(14):945 – 954, 2003. ISSN 0950-5849. doi: 10.1016/s0950-5849(03)00096-x. URL `http://www.sciencedirect.com/science/article/pii/S095058490300096X`. Eighth International Workshop on Requirements Engineering: Foundation for Software Quality.

[23] Banu Aysolmaz. *UPROM: A UNIFIED BUSINESS PROCESS MODELING METHODOLOGY*. PhD thesis, THE GRADUATE SCHOOL OF INFORMATICS OF MIDDLE EAST TECHNICAL UNIVERSITY, 2014.

[24] Banu Aysolmaz. Unified Business Process Modeling Methodology (UPROM) Application: Case Study Results METU/II-TR-2014-32. Technical report, Informatics Institute, METU, Ankara, Turkey, 2014. URL `http://expertjudgment.com/publications/METU_II_TR_2014_32.pdf`.

[25] Banu Aysolmaz and Onur Demirörs. Deriving user requirements from business process models for automation: A case study. In *1st IEEE International Workshop on the Interrelations between Requirements Engineering and Business Process Management, REBPM 2014, Karlskrona, Sweden, August 25, 2014*, pages 19–28, 2014. doi: 10.1109/rebpm.2014.6890732.

[26] Banu Aysolmaz and Onur Demirörs. Modeling business processes to generate artifacts for software development: A methodology. In *Proceedings of the 6th International Workshop on Modeling in Software Engineering*, MiSE 2014, pages 7–12, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2849-4. doi: 10.1145/2593770.2593775.

[27] Banu Aysolmaz and Onur Demirörs. Modeling Business Processes to Generate Artifacts for Software Development : A Methodology. In *Modeling in Software Engineering (MISE), 2014 ICSE Workshop on*, Hydarabad, India, 2014. doi: 10.1145/2593770.2593775.

[28] Zeina Azmeh, Isabelle Mirbel, and Pierre Crescenzo. Highlighting stakeholder communities to support requirements decision-making. In Joerg Doerr and AndreasL. Opdahl, editors, *Requirements Engineering: Foundation for Software Quality*, volume 7830 of *Lecture Notes in Computer Science*, pages 190–205. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-37421-0. doi: 10.1007/978-3-642-37422-7_14.

[29] Ebrahim Bagheri, Mohsen Asadi, Dragan Gasevic, and Samaneh Soltani. Stratified analytic hierarchy process: Prioritization and selection of software features. In Jan Bosch and Jaejoon Lee, editors, *Software Product Lines: Going Beyond*, volume 6287 of *Lecture Notes in Computer Science*, pages 300–315. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-15578-9. doi: 10.1007/978-3-642-15579-6_21.

[30] P. Baker, M. Harman, K. Steinhofel, and A. Skaliotis. Search based approaches to component selection and prioritization for the next release problem. In *Software Maintenance, 2006. ICSM '06. 22nd IEEE International Conference on*, pages 176–185, Sept 2006. doi: 10.1109/icsm.2006.56.

[31] Luciana Ballejos and Jorge Montagna. Method for stakeholder identification in interorganizational environments. *Requirements Engineering*, 13(4):281–297, November 2008. doi: 10.1007/s00766-008-0069-1.

[32] Richard L. Baskerville. Investigating information systems with action research. *Commun. AIS*, 2(3es), November 1999. URL `http://dl.acm.org/citation.cfm?id=374468.374476`.

[33] Peter Baumgartner and Christian Kohls. Finding the "right" level of abstraction for patterns. In *Proceedings of the 20th Conference on Pattern Languages of Programs*, PLoP '13, pages 31:1–31:16, USA, 2013. The Hillside Group. ISBN 978-1-941652-00-8. URL `http://dl.acm.org/citation.cfm?id=2725669.2725706`.

[34] Nigan Bayazit. Investigating Design: A Review of Forty Years of Design Research. *Design Issues*, 20(1,):16–29, 2004. URL `http://www.mitpressjournals.org/doi/abs/10.1162/074793604772933739?prevSearch=authorsfield%3A%28Bayazit%2CNigan%29`.

[35] Guy Beaucamp and Lutz Treder. *Methoden und Techniken der Rechtsanwendung*. C.F.Müller, 2. edition, 2011.

[36] K. Beckers, D. Hatebur, and M. Heisel. A problem-based threat analysis in compliance with common criteria. In *ARES '13*. IEEE Computer Society, 2013. URL `http://www.ieee.org/`.

[37] Kristian Beckers. *pattern- and security-requirements-engineering-based establishment of security standards*. PhD thesis, University Duisburg-Essen, 2014.

[38] Kristian Beckers and Stephan Faßbender. Peer-to-peer driven software engineering considering security, reliability, and performance. In *Availability, Reliability and Security (ARES), 2012 Seventh International Conference on*, pages 485–494. IEEE, 2012. ISBN 978-1-4673-2244-7. doi: 10.1109/ARES.2012.26. URL `http://doi.ieeecomputersociety.org/10.1109/ARES.2012.26`.

[39] Kristian Beckers, Holger Schmidt, Jan-Christoph Küster, and Stephan Faßbender. Pattern-based support for context establishment and asset identification of the iso 27000 in the field of cloud computing. In *Sixth International Conference on Availability, Reliability and Security, ARES 2011*, pages 327–333. IEEE, 2011. ISBN 978-1-4577-0979-1. doi: 10.1109/ARES.2011.55. URL `http://doi.ieeecomputersociety.org/10.1109/ARES.2011.55`.

[40] Kristian Beckers, Stefan Eicker, Stephan Faßbender, Maritta Heisel, Holger Schmidt, and Widura Schwittek. Ontology-based identification of research gaps and immature research areas. In *International Cross-Domain Conference and Workshop on Availability, Reliability, and Security, CD-ARES 2012*, volume 7465 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2012. ISBN 978-1-4673-2244-7. doi: 10.1007/978-3-642-32498-7_1. URL `http://link.springer.com/chapter/10.1007%2F978-3-642-32498-7_1`.

[41] Kristian Beckers, Stephan Faßbender, Maritta Heisel, and Rene Meis. Pattern-based context establishment for service-oriented architectures. In *Software Service and Application Engineering*, volume 7365 of *Lecture Notes in Computer Science*, pages 81–101. Springer, 2012. ISBN 978-3-642-30834-5. doi: 10.1007/978-3-642-30835-2_7. URL `http://link.springer.com/chapter/10.1007%2F978-3-642-30835-2_7`.

[42] Kristian Beckers, Stephan Faßbender, Maritta Heisel, and Rene Meis. A problem-based approach for computer aided privacy threat identification. In *Privacy Technologies and Policy, First Annual Privacy Forum, APF 2012*, volume 8319 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2012. ISBN 978-3-642-54068-4. doi: 10.1007/978-3-642-54069-1_1. URL `http://link.springer.com/chapter/10.1007/978-3-642-54069-1_1`.

[43] Kristian Beckers, Stephan Faßbender, Jan-Christoph Küster, and Holger Schmidt. A pattern-based method for identifying and analyzing laws. In *Requirements Engineering: Foundation for Software Quality - 18th International Working Conference, REFSQ 2012*, volume 7195 of *Lecture Notes in Computer Science*, pages 256–262. Springer, 2012. ISBN 978-3-642-28713-8. doi: 10.1007/978-3-642-28714-5_23. URL `http://link.springer.com/chapter/10.1007%2F978-3-642-28714-5_23`.

[44] Kristian Beckers, Stephan Faßbender, and Holger Schmidt. An integrated method for pattern-based elicitation of legal requirements applied to a cloud computing example. In *Seventh International Conference on Availability, Reliability and Security, Prague, ARES 2012*, pages 463–472. IEEE Computer Society, 2012. ISBN 978-1-4673-2244-7. doi: 10.1109/ares.2012.25. URL `http://doi.ieeecomputersociety.org/10.1109/ARES.2012.25`.

[45] Kristian Beckers, Isabelle Côté, Stephan Faßbender, Maritta Heisel, and Stefan Hofbauer. A pattern-based method for establishing a cloud-specific information security management system. *Requirements Engineering*, 18(4):1–53, November 2013. doi: http://dx.doi.org/10.1007/s00766-013-0174-7. URL `http://link.springer.com/article/10.1007%2Fs00766-013-0174-7`.

[46] Kristian Beckers, Isabelle Côté, Denis Hatebur, Stephan Faßbender, and Maritta Heisel. Common Criteria CompliAnt Software Development (CC-CASD). In *Proceedings 28th Symposium on Applied Computing*, pages 937–943. ACM, 2013. ISBN 978-1-4503-1656-9. doi: 10.1145/2480362.2480604. URL `http://dl.acm.org/citation.cfm?id=2480604`.

[47] Kristian Beckers, Stephan Faßbender, and Maritta Heisel. A meta-model for context-patterns. In *Proceedings of the 18th European Conference on Pattern Languages of Program*, EuroPLoP '13, pages 5:1–5:15, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-3465-5. doi: 10.1145/2739011.2739016. URL `http://doi.acm.org/10.1145/2739011.2739016`.

[48] Kristian Beckers, Stephan Faßbender, Maritta Heisel, and Federika Paci. Combining goal-oriented and problem-oriented requirements engineering methods. In *Proceedings of the International Cross Domain Conference and Workshop 2013 (CD-ARES 2013)*, volume 8127 of *Lecture Notes in Computer Science*, pages 178–194. Springer, 2013. ISBN 978-3-642-40510-5. doi: 10.1007/978-3-642-40511-2_13. URL `http://link.springer.com/chapter/10.1007%2F978-3-642-40511-2_13`.

[49] Kristian Beckers, Stephan Faßbender, and Maritta Heisel. A meta-pattern and pattern form for context-patterns. In *Proceedings of the 19th European Conference on Pattern Languages of Programs (Europlop)*, EuroPLoP '14, pages 5:1–5:23, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3416-7. doi: 10.1145/2721956.2721979. URL `http://doi.acm.org/10.1145/2721956.2721979`.

[50] Kristian Beckers, Stephan Faßbender, and Maritta Heisel. Deriving a pattern language syntax for context-patterns. In *Proceedings of the 19th European Conference on Pattern Languages of Programs (Europlop)*, EuroPLoP '14, pages 2:1–2:25, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3416-7. doi: 10.1145/2721956.2721967. URL `http://doi.acm.org/10.1145/2721956.2721967`.

[51] Kristian Beckers, Stephan Faßbender, Maritta Heisel, and Santiago Suppan. A Threat Analysis Methodology for Smart Home Scenarios. In *Smart Grid Security, Proceddings of the Second International Workshop (SmartGridSec)*, volume 8448 of *Lecture Notes in Computer Science*, pages 94–124. Springer, 2014. ISBN 978-3-319-10328-0. doi: 10.1007/978-3-319-10329-7_7. URL `http://link.springer.com/chapter/10.1007/978-3-319-10329-7_7`.

[52] Tarek Ben Mena, Narjs Bellamine-Ben Saoud, Mohamed Ben Ahmed, and Bernard Pavard. Towards a methodology for context sensitive systems development. In *Modeling and Using Context*, volume 4635 of *Lecture Notes in Computer Science*, pages 56–68. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-74254-8. doi: 10.1007/978-3-540-74255-5_5.

[53] Patrik Berander and Anneliese Andrews. Requirements prioritization. In Aybüke Aurum and Claes Wohlin, editors, *Engineering and Managing Software Requirements*, pages 69–94. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-25043-2. doi: 10.1007/3-540-28244-0_4.

[54] Daniel M. Berry. The importance of ignorance in requirements engineering. *J. Syst. Softw.*, 28(2):179–184, February 1995. ISSN 0164-1212. doi: 10.1016/0164-1212(94)00054-q.

[55] Jean Bézivin. In search of a basic principle for model driven engineering. *Novatica Journal, Special Issue*, 5(2):21–24, 2004.

[56] Jean Bézivin. On the unification power of models. *Software and System Modeling*, 4(2):171–188, 2005. doi: 10.1007/s10270-005-0079-0.

[57] Carola Biagioli, Paola Mariani, and Daniela Tiscornia. Esplex: A rule and conceptual model for representing statutes. In *ICAIL*, pages 240–251. ACM, 1987.

[58] H. C. Black and B. A. Garner. *Black's Law Dictionary*. Black's Law Dictionary Series. West Group, 1999. ISBN 9780314228642. URL `http://books.google.de/books?id=q_BGAAAAMAAJ`.

[59] S. J. Bleistein, K. Cox, and J. Verner. Requirements Engineering for e-Business Systems: Integrating Jackson Problem Diagrams with Goal Modeling and BPM. *11th Asia Pacific Software Engineering Conference, Busan, Korea*, 2004.

[60] Bobs Guide. BPM more important to UK businesses than the rest of Europe, 2006. URL `http://www.bobsguide.com/guide/news/2006/Oct/17/bpm-more-important-to-uk-businesses-than-the-rest-of-europe-but-lags-behind-on-delivery-filenet-survey.html`.

[61] B. W. Boehm and P. N. Papaccio. Understanding and controlling software costs. *IEEE Transactions on Software Engineering*, 14(10):1462–1477, 1988.

[62] Guido Boella, Llio Humphreys, Robert Muthuri, Piercarlo Rossi, and Leendert van der Torre. A critical analysis of legal requirements engineering from the perspective of legal practice. In *RELAW'14*, pages 14–21, 2014.

[63] Christian Braun, Felix Wortmann, Martin Hafner, and Robert Winter. Method construction - a core approach to organizational engineering. In *Proceedings of the 2005 ACM Symposium on Applied Computing*, SAC '05, pages 1295–1299, New York, NY, USA, 2005. ACM. ISBN 1-58113-964-0. doi: 10.1145/1066677.1066971.

[64] Travis Breaux. A method to acquire compliance monitors from regulations. In *Proceedings of the International Workshop on Requirements Engineering and Law (RELAW)*, pages 17–26. IEEE, 2010.

[65] Travis Breaux and Annie I. Antón. Analyzing regulatory rules for privacy and security requirements. *IEEE Transactions on Software Engineering*, 34(1):5–20, 2008.

[66] Travis Breaux, Matthew W. Vail, and Annie I. Antón. Towards regulatory compliance: Extracting rights and obligations to align requirements with regulations. In *Proceedings of the International Conference on Requirements Engineering (RE)*, pages 46–55. IEEE, 2006.

[67] Travis D. Breaux. Exercising due diligence in legal requirements acquisition: A tool-supported, frame-based approach. In *RE 2009, 17th IEEE International Requirements Engineering Conference, Atlanta, Georgia, USA, August 31 - September 4, 2009*, pages 225–230, 2009. doi: 10.1109/re.2009.46.

[68] Travis D. Breaux and David L. Baumer. Legally 'reasonable' security requirements: A 10-year ftc retrospective. *Computers and Security*, 30(4):178–193, 2011.

[69] Travis D. Breaux and David G. Gordon. Regulatory requirements traceability and analysis using semi-formal specifications. In *Requirements Engineering: Foundation for Software Quality - 19th International Working Conference, REFSQ 2013, Essen, Germany, April 8-11, 2013. Proceedings*, pages 141–157, 2013. doi: 10.1007/978-3-642-37422-7_11.

[70] Travis D. Breaux, Annie I. Antón, Kent Boucher, and Merlin Dorfman. Legal requirements, compliance and practice: An industry case study in accessibility. In *RE'08: Proceedings of the 16th IEEE International Requirements Engineering Conference (RE'08)*, pages 43–52, Washington, DC, USA, September 2008. IEEE Society Press.

[71] Roman Brehm. Kryptographische Verfahren in Internetwahlsystemen. Technical report, Technical University of Darmstadt, June 2012.

[72] Pearl Brereton, Barbara Kitchenham, David Budgen, Mark Turner, and Mohamed Khalil. Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software*, 80(4):571–583, 2007. doi: 10.1016/j.jss.2006.07.009.

[73] Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia, and John Mylopoulos. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, May 2004. ISSN 1387-2532. doi: 10.1023/B:AGNT.0000018806.20944.ef. URL `http://dx.doi.org/10.1023/B:AGNT.0000018806.20944.ef`.

[74] Patrick Brézillon. Context in problem solving: A survey. *Knowl. Eng. Rev.*, 14(1):47–80, May 1999. ISSN 0269-8889. doi: 10.1017/S0269888999141018. URL `http://dx.doi.org/10.1017/S0269888999141018`.

[75] Frederick P. Brooks, Jr. *The Mythical Man-month (Anniversary Ed.)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995. ISBN 0-201-83595-9.

[76] BSI. Protection Profile for the Gateway of a Smart Metering System (Gateway PP). Version 01.01.01(final draft), Bundesamt für Sicherheit in der Informationstechnik (BSI) - Federal Office for Information Security Germany, Bonn,Germany, 2011. `https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/SmartMeter/PP-SmartMeter.pdf?__blob=publicationFile`.

[77] BSI. Protection Profile for the Security Module of a Smart Meter Gateway (Security Module PP). Version 1.0), Bundesamt für Sicherheit in der Informationstechnik (BSI) - Federal Office for Information Security Germany, Bonn,Germany, 2013. `https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/SmartMeter/PP_Security_%20Module.pdf?__blob=publicationFile`.

[78] Jim Buchan and Christian Hasana Ekadharmawan. Barriers to sharing domain knowledge in software development practices in smes. In *3rd International Workshop on Knowledge Collaboration in Software Development*, 2009.

[79] Jim Buchan, C. H. Ekadharmawan, and S. G. MacDonell. Insights into domain knowledge sharing in software development practice in smes. In *Software Engineering Conference, 2009. APSEC '09. Asia-Pacific*, pages 93–100, 2009. doi: 10.1109/apsec.2009.47.

[80] David Budgen, Mark Turner, Pearl Brereton, and Barbara Kitchenham. Using Mapping Studies in Software Engineering. In *Proceedings of PPIG 2008*, pages 195–204. Lancaster University, 2008. ISBN 978-1-86220-215-3.

[81] David Budgen, Barbara Kitchenham, and O. Pearl Brereton. Using mapping studies as the basis for further research - a participant-observer case study. *Information & Software Technology*, 53(6):638–651, 2011. doi: 10.1016/j.infsof.2010.12.011.

[82] Jens Budszus, Oliver Berthold, Alexander Filip, Sven Polenz, Thomas Probst, and Maren Thiermann. Orientierungshilfe: Cloud computing. Technical report, Arbeitskreise Technik und Medien der Konferenz der Datenschutzbeauftragten des Bundes und der Länder sowie der Arbeitsgruppe Internationaler Datenverkehr des Düsseldorfer Kreises, 2014. URL `http://www.datenschutz-berlin.de/attachments/1063/2014-Orientierungshilfe_Cloud_Computing.pdf`.

[83] Bundesbeauftragter für den Datenschutz und die Informationsfreiheit. Datenschutz und informationsfreiheit-bericht 2008, 12 2008. URL `http://www.datenschutz-berlin.de/attachments/585/2008-datenschutzbericht_verlinkt_READER_vom_03.04.09.pdf?1241010791,LastAccess:06.10.2014`.

[84] Bundeskriminalamt (federal criminal police office). Bundeslagebild Cybercrime 2012 (report on cybercrime 2012). Technical report, Germany, 2013.

[85] Bundeskriminalamt (federal criminal police office). Bundeslagebild Cybercrime 2013 (report on cybercrime 2013). Technical report, Germany, 2014.

[86] Zmirir Bundestag der Bundesrepublik Deutschland (Lower House of German Parliament). Ladenschlußgesetz (Store-closing Act), October 2006. available at: http://www.gesetze-im-internet.de/bundesrecht/ladschlg/gesamt.pdf.

[87] Zmirir Bundestag der Bundesrepublik Deutschland (Lower House of German Parliament). Telemediengesetz (Tele-media Act), February 2007. available at: http://www.gesetze-im-internet.de/bundesrecht/tmg/gesamt.pdf.

[88] Zmirir Bundestag der Bundesrepublik Deutschland (Lower House of German Parliament). Bundesdatenschutzgesetz (Federal Data Protection Act), August 2009. available at: http://www.gesetze-im-internet.de/englisch_bdsg/federal_data_protection_act.pdf.

[89] Zmirir Bundestag der Bundesrepublik Deutschland (Lower House of German Parliament). Grundgesetz (Basic Law), July 2010. available at: http://www.gesetze-im-internet.de/englisch_gg/basic_law_for_the_federa_republic_of_germany.pdf.

[90] Corentin Burnay, Ivan Jureta, and Stéphane Faulkner. Influence of context on decision making during requirements elicitation. *CoRR*, abs/1210.7101, 2012.

[91] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. *Pattern-Oriented Software Architecture Volume 1: A System of Patterns*. Wiley, 1996.

[92] Frank Buschmann, Kevlin Henney, and Douglas C. Schmidt. *Pattern-Oriented Software Architecture Volume 5: On Patterns and Pattern Languages*. Wiley, 2007.

[93] Muffy Calder, Mario Kolberg, Evan H. Magill, and Stephan Reiff-Marganiec. Feature interaction: a critical review and considered forecast. *Comput. Netw.*, 41:115–141, 2003. ISSN 1389-1286.

[94] E. J. Cameron and H. Velthuijsen. Feature interactions in telecommunications systems. *Comm. Mag.*, 31 (8):18–23, August 1993. ISSN 0163-6804. doi: 10.1109/35.229532. URL `http://dx.doi.org/10.1109/35.229532`.

[95] Pär Carlshamre. Release planning in market-driven software product development: Provoking an understanding. *Requirements Engineering*, 7(3):139–151, 2002. ISSN 0947-3602. doi: 10.1007/s007660200010.

[96] Pär Carlshamre, K. Sandahl, M. Lindvall, B. Regnell, and J. Natt och Dag. An industrial survey of requirements interdependencies in software product release planning. In *Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on*, pages 84–91, 2001. doi: 10.1109/isre.2001.948547.

[97] Carlos Castro-Herrera and Jane Cleland-Huang. A machine learning approach for identifying expert stakeholders. *Managing Requirements Knowledge, International Workshop on*, pages 45–49, 2009. doi: 10.1109/mark.2009.1.

[98] H. Cavusoglu, B. Mishra, and S. Raghunathan. The effect of internet security breach announcements on market value: Capital market reactions for breached firms and internet security developers. *Int. J. Electron. Commerce*, 9(1):70–104, 2004.

[99] P. Checkland. *Systems thinking, systems practice*. J. Wiley, 1981. ISBN 9780471279112. URL `https://books.google.de/books?id=icXaAAAAMAAJ`.

[100] B. H. C. Cheng and J. M. Atlee. Research directions in requirements engineering. In *Future of Software Engineering, 2007. FOSE '07*, pages 285–303, May 2007. doi: 10.1109/fose.2007.17.

[101] Hyun Cho and Jeff Gray. Design patterns for metamodels. In *Proceedings of the Compilation of the Co-located Workshops on DSM'11, TMC'11, AGERE! 2011, AOOPES'11, NEAT'11, & VMIL'11*, pages 25–32, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-1183-0. doi: 10.1145/2095050.2095056.

[102] Commission of the European communities. Communication from the commission to the european parliament, the council, the european economic and social committee and the committee of the regions. Technical report, Commission of the European communities, 2011.

[103] Isabelle Côté, Denis Hatebur, Maritta Heisel, Holger Schmidt, and Ina Wentzlaff. A systematic account of problem frames. In *Proceedings of the European Conference on Pattern Languages of Programs (EuroPLoP)*, pages 749–767. Universitätsverlag Konstanz, 2008.

[104] Isabelle Côté, Denis Hatebur, Maritta Heisel, and Holger Schmidt. UML4PF – a tool for problem-oriented requirements analysis. In *Proceedings of the International Conference on Requirements Engineering (RE)*, pages 349–350. IEEE Computer Society, 2011.

[105] Jane Coughlan and Robert D. Macredie. Effective communication in requirements elicitation: A comparison of methodologies. *Requirements Engineering*, 7(2):47–60, 2002. ISSN 0947-3602. doi: 10.1007/s007660200004.

[106] Jane Coughlan, Mark Lycett, and Robert D. Macredie. Communication issues in requirements elicitation: a content analysis of stakeholder experiences. *Information and Software Technology*, 45(8):525 – 537, 2003. ISSN 0950-5849. doi: 10.1016/s0950-5849(03)00032-6. URL `http://www.sciencedirect.com/science/article/pii/S0950584903000326`.

[107] Nigel Cross. *Engineering Design Methods: Strategies for Product Design (4th ed.)*. John Wiley & Sons, Chichester, April 2008.

[108] Bill Curtis, Herb Krasner, and Neil Iscoe. A field study of the software design process for large systems. *Commun. ACM*, 31(11):1268–1287, 1988. ISSN 0001-0782. doi: 10.1145/50087.50089.

[109] Daniela Damian, Remko Helms, Irwin Kwan, Sabrina Marczak, and Benjamin Koelewijn. The role of domain knowledge and cross-functional communication in socio-technical coordination. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pages 442–451, Piscataway, NJ, USA, 2013. IEEE Press. ISBN 978-1-4673-3076-3. URL `http://dl.acm.org/citation.cfm?id=2486788.2486847`.

[110] M. Daneva and A. Herrmann. Requirements prioritization based on benefit and cost prediction: A method classification framework. In *Software Engineering and Advanced Applications, 2008. SEAA '08. 34th Euromicro Conference*, pages 240–247, Sept 2008. doi: 10.1109/seaa.2008.46.

[111] Robert Darimont and Michel Lemoine. Goal-oriented analysis of regulations. In *Proceedings of the CAISE*06 Workshop on Regulations Modelling and their Validation and Verification ReMo2V '06, Luxemburg, June 5-9, 2006*, 2006. URL `http://ceur-ws.org/Vol-241/paper9.pdf`.

[112] A. Davis, O. Dieste, A. Hickey, N. Juristo, and A. M. Moreno. Effectiveness of requirements elicitation techniques: Empirical results derived from a systematic review. In *Requirements Engineering, 14th IEEE International Conference*, pages 179–188, Sept 2006. doi: 10.1109/RE.2006.17.

[113] Robert M. Davison, Maris G. Martinsons, and Ned Kock. Principles of canonical action research. *Inf. Syst. J.*, 14(1):65–, 2004. doi: 10.1111/j.1365-2575.2004.00162.x.

[114] O. Demirörs, C. Gencel, and A. Tarhan. Utilizing Business Process Models for Requirements Elicitation. In *Proceedings of the 29th EUROMICRO Conference "New Waves in System Architecture" (EUROMICRO'03)*, 2003.

[115] Department of Energy and Climate Change. Smart Metering Implementation Programme, Response to Prospectus Consultation, Overview Document. Technical report, Office of Gas and Electricity Markets, 2011.

[116] Department of Energy and Climate Change. Smart Metering Implementation Programme, Response to Prospectus Consultation, Design Requirements. Technical report, Office of Gas and Electricity Markets, 2011.

[117] Anind K. Dey. Understanding and using context. *Personal Ubiquitous Comput.*, 5(1):4–7, January 2001. ISSN 1617-4909. doi: 10.1007/s007790170019.

[118] Linda Dickens and Karen Watkins. Action research: Rethinking lewin. *Management Learning*, (30), 1999. ISSN 1350-5076.

[119] Remco M. Dijkman and Marlon Dumas. Service-oriented design: A multi-viewpoint approach. *International Journal on Cooperative Information Systems*, 13(4), 2004.

[120] D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–207, 1983.

[121] J. B. Dreger. *Function Point Analysis*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989. ISBN 0-13-332321-8.

[122] M Dumas, W Van der Aalst, and A ter Hofstede. *Process-aware information systems : bridging people and software through process technology*. John Wiley & Sons, New Jersey, 2005. ISBN 9780471663065. URL `http://onlinelibrary.wiley.com/doi/10.1002/0471741442.fmatter/summary`.

[123] M Dumas, M La Rosa, J Mendling, and HA Reijers. *Fundamentals of business process management*. Springer, 2013. ISBN 9783642331428. URL `http://link.springer.com/content/pdf/10.1007/978-3-642-33143-5.pdf`.

[124] A. Egyed and P. Grünbacher. Identifying requirements conflicts and cooperation: How quality attributes and automated traceability can help. *IEEE Softw.*, 21(6):50–58, November 2004. ISSN 0740-7459.

[125] Golnaz Elahi and Eric Yu. Requirements trade-offs analysis in the absence of quantitative measures: A heuristic method. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, SAC '11, pages 651–658, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0113-8. doi: 10.1145/1982185.1982331.

[126] Veli-Pekka Eloranta, Johannes Koskinen, Marko Leppänen, and Ville Reijonen. *Designing Distributed Control Systems: A Pattern Language Approach*. Wiley, 2014.

[127] Mark Endrei, Jenny Ang, A. Arsanjani, Sook Chua, Philippe Comte, Pœl Krogdahl, Min Luo, and Tony Newling. *Patterns: Service-Oriented Architecture and Web Services*. IBM Redbooks, July 2004. URL `http://www.redbooks.ibm.com/redbooks/pdfs/sg246303.pdf`.

[128] Wilco Engelsman, Dick Quartel, Henk Jonkers, and Marten van Sinderen. Extending enterprise architecture modelling with business goals and requirements. *Enterprise Information Systems*, 5(1):9–36, February 2011. URL `http://doc.utwente.nl/75669/`.

[129] K. Engisch, T. Würtenberger, and D. Otto. *Einführung in das juristische Denken*. Kohlhammer-Urban-Taschenbücher. Kohlhammer, 2005. ISBN 9783170186958.

[130] Sergio Espana, Arturo Gonzalez, and Oscar Pastor. Communication analysis: A requirements engineering method for information systems. In Pascal van Eck, Jaap Gordijn, and Roel Wieringa, editors, *Advanced Information Systems Engineering*, volume 5565 of *Lecture Notes in Computer Science*, pages 530–545. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-02143-5. doi: 10.1007/978-3-642-02144-2_41. URL `http://dx.doi.org/10.1007/978-3-642-02144-2_41`.

[131] Benjamin Fabian, Seda Gürses, Maritta Heisel, Thomas Santen, and Holger Schmidt. A comparison of security requirements engineering methods. *Requirements Engineering – Special Issue on Security Requirements Engineering*, 15(1):7–40, 2010.

[132] Stephan Faßbender and Banu Aysolmaz. Transforming process models to problem frames. LNBIB. Springer, 2015.

[133] Stephan Faßbender and Maritta Heisel. From problems to laws in requirements engineering using model-transformation (best students paper award). In *ICSOFT 2013 - Proceedings of the 8th International Conference on Software Paradigm Trends*, pages 447–458. SciTePress, 2013. ISBN 978-989-8565-68-6. doi: 10.5220/0004490804470458. URL `http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0004490804470458`.

[134] Stephan Faßbender and Maritta Heisel. A computer aided process from problems to laws in requirements engineering. In *Software Technologies*, volume 457 of *Communications in Computer and Information Science*, pages 215–234. Springer, 2014. ISBN 978-3-662-44919-6. doi: 10.1007/978-3-662-44920-2_14. URL `http://link.springer.com/chapter/10.1007%2F978-3-662-44920-2_14`.

[135] Stephan Faßbender, Maritta Heisel, and Rene Meis. Functional requirements under security pressure (best students paper award). In *ICSOFT-PT 2014 - Proceedings of the 9th International Conference on Software Paradigm Trends, Vienna, Austria, 29-31 August, 2014*, pages 5–16. SciTePress, 2014. ISBN 978-989-758-036-9. doi: 10.5220/0005098600050016. URL `http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0005098600050016`.

[136] Stephan Faßbender, Maritta Heisel, and Rene Meis. Aspect-oriented requirements engineering with problem frames. In *ICSOFT-PT 2014 - Proceedings of the 9th International Conference on Software Paradigm Trends, Vienna, Austria, 29-31 August, 2014*, pages 145–156. SciTePress, 2014. ISBN 978-989-758-036-9. doi: 10.5220/0005001801450156. URL `http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0005001801450156`.

[137] Stephan Faßbender, Maritta Heisel, and Rene Meis. Problem-based security requirements elicitation and refinement with pressure. In *Software Technologies*, volume - of *Communications in Computer and Information Science*, pages –. Springer, 2015. ISBN -. doi: -. URL -.

[138] Stephan Faßbender, Maritta Heisel, and Rene Meis. A problem-, quality-, and aspect-oriented requirements engineering method. In *Software Technologies*, volume - of *Communications in Computer and Information Science*, pages –. Springer, 2015. ISBN -. doi: -. URL -.

[139] M. S. Feather, S. L. Cornford, K. A. Hicks, J. D. Kiper, and T. Menzies. A broad, quantitative model for making early requirements decisions. *Software, IEEE*, 25(2):49–56, March 2008. ISSN 0740-7459. doi: 10.1109/ms.2008.29.

[140] Federal Constitutional Court of Germany. Verwendung von Wahlcomputern bei der Bundestagswahl 2005 verfassungswidrig, March 2009. URL `https://www.bundesverfassungsgericht.de/pressemitteilungen/bvg09-019.html`. https://www.bundesverfassungsgericht.de/pressemitteilungen/bvg09-019.html.

[141] Federal Trade Commission. Choicepoint settles data security breach charges. Technical report, Federal Trade Commission, 2006. Available at `http://www.ftc.gov/opa/2006/01/choicepoint.shtm`.

[142] Norman E. Fenton and Shari Lawrence Pfleeger. *Software Metrics: A Rigorous and Practical Approach*. PWS Publishing Co., Boston, MA, USA, 2nd edition, 1998. ISBN 0534954251.

[143] Eduardo B. Fernandez and Rouyi Pan. A Pattern Language for Security Models. In *8th Conference of Pattern Languages of Programs (PloP)*, 2001.

[144] Anthony Finkelstein, Mark Harman, S. Afshin Mansouri, Jian Ren, and Yuanyuan Zhang. A search based approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making. *Requirements Engineering*, 14(4):231–245, 2009. ISSN 0947-3602. doi: 10.1007/s00766-009-0075-y.

[145] D. Firesmith. Specifying good requirements. *Journal of Object Technology*, 2(4), 2003.

[146] Donald Firesmith. Prioritizing requirements. *Journal of Object Technology*, 3(8):35–48, 2004.

[147] Martin Fowler. *Analysis Patterns: Reusable Object Models*. Addison-Wesley, 1996.

[148] Martin Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002. ISBN 0321127420.

[149] Robert France and Bernhard Rumpe. Model-driven development of complex software: A research roadmap. In *2007 Future of Software Engineering*, FOSE '07, pages 37–54, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2829-5. doi: 10.1109/fose.2007.14.

[150] Xavier Franch. On the quantitative analysis of agent-oriented models. In Eric Dubois and Klaus Pohl, editors, *Advanced Information Systems Engineering*, volume 4001 of *Lecture Notes in Computer Science*, pages 495–509. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-34652-4. doi: 10.1007/11767138_33. URL `http://dx.doi.org/10.1007/11767138_33`.

[151] Ganna Frankova, Magali Seguran, Florian Gilcher, Slim Trabelsi, Joerg Doerflinger, and Marco Aiello. Deriving business processes with service level agreements from early requirements. *Journal of systems and software*, 84(8):1351–1363, 2011. ISSN 0164-1212. doi: 10.1016/j.jss.2011.03.077. Relation: http://www.rug.nl/fmns-research/bernoulli/index Rights: University of Groningen, Johann Bernoulli Institute for Mathematics and Computer Science.

[152] R.E. Freeman. *Strategic Management: A Stakeholder Approach*. Pitman Series in Business and Public Policy. Cambridge University Press, 2010. ISBN 9780521151740. URL `https://books.google.co.uk/books?id=NpmA_qEiOpkC`.

[153] Ruben Fuentes-Fernandez, JorgeJ. Gomez-Sanz, and Juan Pavon. Understanding the human context in requirements elicitation. *Requirements Engineering*, 15(3):267–283, 2010. ISSN 0947-3602. doi: 10.1007/s00766-009-0087-7.

[154] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

[155] E. Garcia Alcazar and A. Monzon. A process framework for requirements analysis and specification. In *Requirements Engineering, 2000. Proceedings. 4th International Conference on*, pages 27–35, 2000. doi: 10.1109/icre.2000.855582.

[156] Sepideh Ghanavati, Daniel Amyot, and Liam Peyton. A requirements management framework for privacy compliance. In *Anais do WER07 - Workshop em Engenharia de Requisitos, Toronto, Canada, May 17-18, 2007*, pages 149–159, 2007. URL `http://wer.inf.puc-rio.br/WERpapers/artigos/artigos_WER07/Qwer07-ghanavati.pdf`.

[157] Sepideh Ghanavati, Daniel Amyot, and Liam Peyton. Compliance analysis based on a goal-oriented requirement language evaluation methodology. In *RE 2009, 17th IEEE International Requirements Engineering Conference, Atlanta, Georgia, USA, August 31 - September 4, 2009*, pages 133–142, 2009. doi: 10.1109/re.2009.42.

[158] B. Glaser and A. Strauss. *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Aldine Transaction, 1967.

[159] D. G. Gordon and T. D. Breaux. Comparing requirements from multiple jurisdictions. In *Requirements Engineering and Law (RELAW), 2011 Fourth International Workshop on*, pages 43–49, Aug 2011. doi: 10.1109/relaw.2011.6050272.

[160] David G. Gordon and Travis D. Breaux. A cross-domain empirical study and legal evaluation of the requirements water marking method. *Requirements Engineering*, 18(2):147–173, 2013.

[161] David G. Gordon and Travis D. Breaux. Assessing regulatory change through legal requirements coverage modeling. In *21st IEEE International Requirements Engineering Conference, RE 2013, Rio de Janeiro-RJ, Brazil, July 15-19, 2013*, pages 145–154, 2013. doi: 10.1109/re.2013.6636714.

[162] D Greer and G Ruhe. Software release planning: an evolutionary and iterative approach. *Information and Software Technology*, 46(4):243 – 253, 2004. ISSN 0950-5849. doi: 10.1016/j.infsof.2003.07.002. URL `http://www.sciencedirect.com/science/article/pii/S095058490300140X`.

[163] Gerd Gröner, Mohsen Asadi, Bardia Mohabbati, Dragan Gasevic, Fernando Silva Parreiras, and Marko Boskovic. Validation of User Intentions in Process Models. In *Advanced Information Systems Engineering - 24th International Conference, CAiSE*, volume 7328 of *LNCS*, pages 366–381. Springer, 2012. doi: 10.1007/978-3-642-31095-9_24.

[164] Object Management Group. Service oriented architecture modeling language 1.0.1. Technical report, Object Management Group, May 2012. URL `http://www.omg.org/spec/SoaML/1.0.1/PDF`.

[165] Irit Hadar, Pnina Soffer, and Keren Kenzi. The role of domain knowledge in requirements elicitation via interviews: an exploratory study. *Requirements Engineering*, 19(2):143–159, 2014. ISSN 0947-3602. doi: 10.1007/s00766-012-0163-2.

[166] Munawar Hafiz, Paul Adamczyk, and Ralph E. Johnson. Growing a pattern language (for security). In *Proceedings of the ACM international symposium on New ideas, new paradigms, and reflections on programming and software*, Onward! '12, pages 139–158, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1562-3.

[167] C. B. Haley, R. Laney, J. D. Moffett, and B. Nuseibeh. Security requirements engineering: A framework for representation and analysis. *IEEE Transactions on Software Engineering*, 34(1):133–153, 2008.

[168] A. D. Hall. *A methodology for systems engineering.* Van Nostrand, 1962.

[169] T. Hall, S. Beecham, and A. Rainer. Requirements problems in twelve software companies: an empirical analysis. *Software, IEE Proceedings* -, 149(5):153–160, Oct 2002. ISSN 1462-5970. doi: 10.1049/ip-sen:20020694.

[170] W. Hassan and Luigi Logrippo. Requirements and compliance in legal systems: a logic approach. In *Requirements Engineering and Law, 2008. RELAW '08.*, pages 40–44, Sept 2008. doi: 10.1109/relaw.2008.8.

[171] Waël Hassan and Luigi Logrippo. Governance requirements extraction model for legal compliance validation. In *Proceedings of the International Workshop on Requirements Engineering and Law (RELAW)*, pages 7–12. IEEE Computer Society, 2009.

[172] Waël Hassan and Luigi Logrippo. Towards a process for legally compliant software. *2013 6th International Workshop on Requirements Engineering and Law (RELAW)*, 0:44–52, 2013. doi: 10.1109/relaw.2013.6671345.

[173] D. Hatebur and M. Heisel. A Foundation for Requirements Analysis of Dependable Software. In B. Buth, G. Rabe, and T. Seyfarth, editors, *Proc. of the Int. Conf. on Computer Safety, Reliability and Security (SAFECOMP)*, LNCS 5775, pages 311–325. Springer, 2009.

[174] D. Hatebur and M. Heisel. A UML Profile for Requirements Analysis of Dependable Software. In E. Schoitsch, editor, *Proc. of the Int. Conf. on Computer Safety, Reliability and Security (SAFECOMP)*, LNCS 6351, pages 317–331. Springer, 2010. ISBN 978-3-642-15650-2.

[175] Denis Hatebur. *Pattern- and Component-based Development of Dependable Systems.* PhD thesis, 2012. URL `http://www.dwverlag.de/index.php?art=Pattern-+and+Component-based+Development+of+Dependable+Systems&mod=Onlineshop&view=Artikel&abid=166`.

[176] Denis Hatebur and Maritta Heisel. Making pattern- and model-based software development more rigorous. In *Proceedings of 12th International Conference on Formal Engineering Methods, ICFEM 2010, Shanghai, China*, LNCS 6447, pages 253–269. Springer, 2010. URL `http://www.springerlink.com/`.

[177] W. Heaven and E. Letier. Simulating and optimising design decisions in quantitative goal models. In *Requirements Engineering Conference (RE), 2011 19th IEEE International*, pages 79–88, Aug 2011. doi: 10.1109/re.2011.6051653.

[178] Maritta Heisel. Agendas – a concept to guide software development activites. In *Proc. Systems Implementation 2000*, pages 19–32. Chapman & Hall London, 1998.

[179] Brian Henderson-Sellers. *On the Mathematics of Modelling, Metamodelling, Ontologies and Modelling Languages*. Springer Publishing Company, Incorporated, 2012. ISBN 3642298249, 9783642298240.

[180] Brian Henderson-Sellers and Cesar Gonzalez-Perez. Granularity in conceptual modelling: Application to metamodels. In Jeffrey Parsons, Motoshi Saeki, Peretz Shoval, Carson C. Woo, and Yair Wand, editors, *ER*, volume 6412 of *Lecture Notes in Computer Science*, pages 219–232. Springer, 2010. ISBN 978-3-642-16372-2. doi: 10.1007/978-3-642-16373-9_16.

[181] Kevlin Henney. Context encapsulation - three stories, a language, and some sequences. In Andy Longshaw and Uwe Zdun, editors, *EuroPLoP*, pages 379–414. UVK - Universitaetsverlag Konstanz, 2005. ISBN 978-3-87940-805-4.

[182] C. Hentrich and Uwe Zdun. A pattern language for process execution and integration design in service-oriented architectures. *Transactions on Pattern Languages of Programming*, 1:136–191, 2009.

[183] A. Herrmann and M. Daneva. Requirements prioritization based on benefit and cost prediction: An agenda for future research. In *International Requirements Engineering, 2008. RE '08. 16th IEEE*, pages 125–134, Sept 2008. doi: 10.1109/re.2008.48.

[184] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS Q.*, 28(1):75–105, March 2004. ISSN 0276-7783. URL `http://dl.acm.org/citation.cfm?id=2017212.2017217`.

[185] H. F. Hofmann and F. Lehner. Requirements engineering as a success factor in software projects. *Software, IEEE*, 18(4):58–66, Jul 2001. ISSN 0740-7459. doi: 10.1109/ms.2001.936219.

[186] Wesley N. Hohfeld. Fundamental legal conceptions as applied in judicial reasoning. *The Yale Law Journal*, 26(8):710–770, 1917.

[187] J. Horkoff, R. Salay, M. Chechik, and A. Di Sandro. Supporting early decision-making in the presence of uncertainty. In *Requirements Engineering Conference (RE), 2014 IEEE 22nd International*, pages 33–42, Aug 2014. doi: 10.1109/re.2014.6912245.

[188] Jennifer Horkoff and Eric Yu. Analyzing goal models: Different approaches and how to choose among them. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, SAC '11, pages 675–682, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0113-8. doi: 10.1145/1982185.1982334. URL `http://doi.acm.org/10.1145/1982185.1982334`.

[189] M. Howard and S. Lipner. *The Security Development Lifecycle : SDL : A Process for Developing Demonstrably More Secure Software*. Microsoft Press, 2006.

[190] John Hutchinson, Mark Rouncefield, and Jon Whittle. Model-driven engineering practices in industry. In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, pages 633–642, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0445-0. doi: 10.1145/1985793.1985882.

[191] IBM. The IBM vision of a smarter home enabled by cloud technology, 2010. URL `http://www.ibm.com/smarterplanet/global/files/uk__uk_en__cloud__a_smarter_home_enabled_by_cloud_computing.pdf?met=uk_smarterplanet_cloud_computing_ideas`.

[192] H. P. In, D. Olson, and T. Rodgers. Multi-criteria preference analysis for systematic requirements negotiation. In *Computer Software and Applications Conference, 2002. COMPSAC 2002. Proceedings. 26th Annual International*, pages 887–892, 2002. doi: 10.1109/cmpsac.2002.1045118.

[193] Marta Indulska, Peter F. Green, Jan Recker, and Michael Rosemann. Business process modeling: Perceived benefits. In *Conceptual Modeling - ER 2009, 28th International Conference on Conceptual Modeling, Gramado, Brazil, November 9-12, 2009. Proceedings*, pages 458–471, 2009. doi: 10.1007/978-3-642-04840-1_34.

[194] Silvia Ingolfo, Alberto Siena, Ivan Jureta, Angelo Susi, Anna Perini, and John Mylopoulos. Choosing compliance solutions through stakeholder preferences. In Joerg Doerr and AndreasL. Opdahl, editors, *Requirements Engineering: Foundation for Software Quality*, volume 7830 of *Lecture Notes in Computer Science*, pages 206–220. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-37421-0. doi: 10.1007/978-3-642-37422-7_15.

[195] International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC). Common Evaluation Methodology 3.1. ISO/IEC 15408, 2009.

[196] Fuyuki Ishikawa, Rihoko Inoue, and Shinichi Honiden. Modeling, analyzing and weaving legal interpretations in goal-oriented requirements engineering. In *Proceedings of the International Workshop on Requirements Engineering and Law (RELAW)*, pages 39–44. IEEE Computer Society, 2009.

[197] Shareeful Islam, Haralambos Mouratidis, and Stefan Wagner. Towards a framework to elicit and manage security and privacy requirements from laws and regulations. In *Requirements Engineering: Foundation for Software Quality, 16th International Working Conference, REFSQ 2010, Essen, Germany, June 30 - July 2, 2010. Proceedings*, pages 255–261, 2010. doi: 10.1007/978-3-642-14192-8_23.

[198] ISO/IEC. Common Criteria for Information Technology Security Evaluation. ISO/IEC 15408, International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), Geneva ,Switzerland, 2009.

[199] ISO/IEC. Information technology - Security techniques - Information security management systems - Overview and Vocabulary. ISO/IEC 27000, International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), Geneva ,Switzerland, 2009.

[200] M. Jackson. *Problem Frames. Analyzing and structuring software development problems.* Addison-Wesley, 2001.

[201] M. Jackson and P. Zave. Domain descriptions. In *International Symposium on Requirements Engineering*, pages 56–64, 1993. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=324836`.

[202] Ivar Jacobson, Grady Booch, and James Rumbaugh. *The Unified Software Development Process.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999. ISBN 0-201-57169-2.

[203] Samireh Jalali and Claes Wohlin. Systematic literature studies: Database searches vs. backward snowballing. In *Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '12, pages 29–38, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1056-7. doi: 10.1145/2372251.2372257.

[204] Pooyan Jamshidi and Claus Pahl. Business process and software architecture model co-evolution patterns. Zurich, Switzerland, 06/2012 2012. doi: org/10.1109/mise.2012.6226021. URL `http://ulir.ul.ie/handle/10344/2701`.

[205] Ho-Won Jung. Optimizing value and cost in requirements analysis. *Software, IEEE*, 15(4):74–78, Jul 1998. ISSN 0740-7459. doi: 10.1109/52.687950.

[206] I. J. Jureta, J. Mylopoulos, and S. Faulkner. Revisiting the core ontology and problem in requirements engineering. In *International Requirements Engineering, 2008. RE '08. 16th IEEE*, pages 71–80, Sept 2008. doi: 10.1109/re.2008.13.

[207] J. Jürjens. *Secure Systems Development with UML.* Springer, 2005.

[208] Erik Kamsties, Daniel M. Berry, Barbara Paech, E. Kamsties, D. M. Berry, and B. Paech. Detecting ambiguities in requirements documents using inspections. In *in Proceedings of the First Workshop on Inspection in Software Engineering (WISE'01*, pages 68–80, 2001.

[209] J. Karlsson and K. Ryan. A cost-value approach for prioritizing requirements. *Software, IEEE*, 14(5):67–74, Sep 1997. ISSN 0740-7459. doi: 10.1109/52.605933.

[210] Joachim Karlsson, Claes Wohlin, and Björn Regnell. An evaluation of methods for prioritizing software requirements. *Information and Software Technology*, 39:939 – 947, 1998. ISSN 0950-5849. doi: 10.1016/s0950-5849(97)00053-0. URL `http://www.sciencedirect.com/science/article/pii/S0950584997000530`.

[211] Gabor Karsai, Holger Krahn, Claas Pinkernell, Bernhard Rumpe, Martin Schindler, and Steven Völkel. Design guidelines for domain specific languages. *CoRR*, abs/1409.2378, 2014.

[212] Ateeq Khan, Christian Kästner, Veit Köppen, and Gunter Saake. Service variability patterns. In *Proceedings of the 30th International Conference on Advances in Conceptual Modeling: Recent Developments and New Directions*, ER'11, pages 130–140, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-24573-2. URL `http://dl.acm.org/citation.cfm?id=2075202.2075225`.

[213] L. Khansa, D. F. Cook, T. James, and O. Bruyaka. Impact of HIPAA provisions on the stock market value of healthcare institutions, and information security and other information technology firms. *Computers & Security*, 31(6):750 – 770, 2012.

[214] Barbara Kitchenham. Procedures for performing systematic reviews. Technical report, Keele University and NICTA, 2004.

[215] Barbara Kitchenham. Template for a Mapping Study Protocol. Technical report, Keele University and NICTA, 2009.

[216] Barbara Kitchenham, Stuart Charters, David Budgen, Pearl Brereton, Mark Turner, Steve Linkman, Magne Jørgensen, Emilia Mendes, and Giuseppe Visaggio. Guidelines for performing systematic literature reviews in software engineering. Technical report, EBSE, 2007.

[217] Barbara Kitchenham, Pearl Brereton, Mark Turner, Mahmood Niazi, Stephen G. Linkman, Rialette Pretorius, and David Budgen. Refining the systematic literature review process - two participant-observer case studies. *Empirical Software Engineering*, 15(6):618–653, 2010. doi: 10.1007/s10664-010-9134-8.

[218] Barbara Kitchenham, Rialette Pretorius, David Budgen, Pearl Brereton, Mark Turner, Mahmood Niazi, and Stephen G. Linkman. Systematic literature reviews in software engineering - a tertiary study. *Information & Software Technology*, 52(8):792–805, 2010. doi: 10.1016/j.infsof.2010.03.006.

[219] Nadzeya Kiyavitskaya, Alzbeta Krausová, and Nicola Zannone. Why Eliciting and Managing Legal Requirements Is Hard. volume 0, pages 26–30, Los Alamitos, CA, USA, September 2008. IEEE Computer Society. ISBN 978-0-7695-3630-9. doi: 10.1109/relaw.2008.10.

[220] Nadzeya Kiyavitskaya, Nicola Zeni, Travis D. Breaux, Annie I. Antón, James R. Cordy, Luisa Mich, and John Mylopoulos. Automating the extraction of rights and obligations for regulatory compliance. In *Conceptual Modeling - ER 2008, 27th International Conference on Conceptual Modeling, Barcelona, Spain, October 20-24, 2008. Proceedings*, pages 154–168, 2008. doi: 10.1007/978-3-540-87877-3_13.

[221] Thorsten Koch. *Rapid Mathematical Programming*. PhD thesis, Technische Universität Berlin, 2004. URL `http://opus4.kobv.de/opus4-zib/frontdoor/index/index/docId/834`. ZIB-Report 04-58.

[222] Ned Kock, Jacques Verville, Azim Danesh-Pajou, and Dorrie DeLuca. Communication flow orientation in business process modeling and its effect on redesign success: Results from a field study. *Decision Support Systems*, 46(2):562–575, January 2009. ISSN 01679236. doi: 10.1016/j.dss.2008.10.002. URL `http://linkinghub.elsevier.com/retrieve/pii/S0167923608001802`.

[223] Christian Kohls. The structure of patterns: Part ii - qualities. In *Proceedings of the 18th Conference on Pattern Languages of Programs*, PLoP '11, pages 27:1–27:18, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-1283-7. doi: 10.1145/2578903.2601079.

[224] Christian Kohls. The path to patterns: Introducing the path metaphor. In *Proceedings of the 17th European Conference on Pattern Languages of Programs*, EuroPLoP '12, pages 9:1–9:16, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-2943-9. doi: 10.1145/2602928.2603085.

[225] Christian Kohls and Katharina Scheiter. The relation between design patterns and schema theory. In *Proceedings of the 15th Conference on Pattern Languages of Programs*, PLoP '08, pages 15:1–15:16, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-151-4. doi: 10.1145/1753196.1753214.

[226] Philippe Kruchten. *The Rational Unified Process: An Introduction*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 3 edition, 2003. ISBN 0321197704.

[227] N. Kukreja, S. S. Payyavula, B. Boehm, and S. Padmanabhuni. Selecting an appropriate framework for value-based requirements prioritization. In *Requirements Engineering Conference (RE), 2012 20th IEEE International*, pages 303–308, Sept 2012. doi: 10.1109/re.2012.6345819.

[228] Sanjay Kumar and Ekta Walia. Analysis of electronic voting system in various countries. *International Journal on Computer Science and Engineering (IJCSE)*, 3(5):18025–1830, May 2011.

[229] A. Lamsweerde. Reasoning about alternative requirements options. In A. Borgida, V. Chaudhri, P. Giorgini, and E. Yu, editors, *Conceptual Modeling: Foundations and Applications*, LNCS 5600, pages 380–397. Springer, 2009.

[230] Karl Larenz. *Methodenlehre der Rechtswissenschaft*. Springer, 5. edition, 1983.

[231] Francis Lau. Toward a framework for action research in information systems studies. *Information Technology & People*, 12(2):148–176, 1999.

[232] P. Laurent, J. Cleland-Huang, and Chuan Duan. Towards automated requirements triage. In *Requirements Engineering Conference, 2007. RE '07. 15th IEEE International*, pages 131–140, Oct 2007. doi: 10.1109/re.2007.63.

[233] Laura Lehtola and Marjo Kauppinen. Suitability of requirements prioritization methods for market-driven software product development. *Software Process: Improvement and Practice*, 11(1):7–19, 2006. ISSN 1099-1670. doi: 10.1002/spip.249.

[234] Laura Lehtola, Marjo Kauppinen, and Sari Kujala. Requirements prioritization challenges in practice. In Frank Bomarius and Hajimu Iida, editors, *Product Focused Software Process Improvement*, volume 3009 of *Lecture Notes in Computer Science*, pages 497–508. Springer Berlin Heidelberg, 2004. ISBN 978-3-540-21421-2. doi: 10.1007/978-3-540-24659-6_36.

[235] Laura Lehtola, Marjo Kauppinen, Jarno Vähäniitty, and Marko Komssi. Linking business and requirements engineering: is solution planning a missing activity in software product companies? *Requir. Eng.*, 14(2): 113–128, 2009. doi: 10.1007/s00766-009-0078-8.

[236] Chiara Leonardi, Luca Sabatucci, Angelo Susi, and Massimo Zancanaro. Ahab's leg: Exploring the issues of communicating semi-formal requirements to the final users. In *Advanced Information Systems Engineering*, volume 6051 of *Lecture Notes in Computer Science*, pages 455–469. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-13093-9. doi: 10.1007/978-3-642-13094-6_35.

[237] Emmanuel Letier, David Stefan, and Earl T. Barr. Uncertainty, risk, and information value in software requirements and architecture. In *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, pages 883–894, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2756-5. doi: 10.1145/2568225.2568239.

[238] Kurt Lewin. Frontiers in Group Dynamics: Concept, Method and Reality in Social Science; Social Equilibria and Social Change. *Human Relations*, 1(1):5–41, 1947. doi: 10.1177/001872674700100103. URL `http://hum.sagepub.com.arugula.cc.columbia.edu:2048/content/vol1/issue1/`.

[239] Chen Li, Marjan van den Akker, Sjaak Brinkkemper, and Guido Diepen. An integrated approach for requirement selection and scheduling in software release planning. *Requirements Engineering*, 15(4):375–396, 2010. ISSN 0947-3602. doi: 10.1007/s00766-010-0104-x.

[240] S. Liaskos, A. Lapouchnian, Yijun Yu, E. b Yu, and J. Mylopoulos. On goal-based variability acquisition and analysis. In *Requirements Engineering, 14th IEEE International Conference*, pages 79–88, Sept 2006. doi: 10.1109/re.2006.45.

[241] Sotirios Liaskos, SheilaA. McIlraith, Shirin Sohrabi, and John Mylopoulos. Representing and reasoning about preferences in requirements engineering. *Requirements Engineering*, 16(3):227–249, 2011. ISSN 0947-3602. doi: 10.1007/s00766-011-0129-9.

[242] Philip Liew, Kostas Kontogiannis, and Tack Tong. A framework for business model driven development. In *STEP*, pages 47–56. IEEE Computer Society, 2004. ISBN 0-7695-2293-9. URL `http://doi.ieeecomputersociety.org/10.1109/STEP.2004.5`.

[243] L. Liu and Z. Jin. Integrating goals and problem frames in requirements analysis. In *RE '06*, pages 349–350, 2006. doi: 10.1109/RE.2006.34.

[244] Lin Liu, Tong Li, and Fei Peng. Why requirements engineering fails: A survey report from china. In *Requirements Engineering Conference (RE), 2010 18th IEEE International*, pages 317–322, Sept 2010. doi: 10.1109/re.2010.45.

[245] Mass Soldal Lund, Bjørnar Solhaug, and Ketil Stolen. *Model-Driven Risk Analysis - The CORAS Approach*. Springer, 2011. ISBN 978-3-642-12322-1. doi: 10.1007/978-3-642-12323-8.

[246] Ioanna Lytra, Stefan Sobernig, Huy Tran, and Uwe Zdun. A pattern language for service-based platform integration and adaptation. In *Proceedings of the 16th European Conference on Pattern Languages of Programs (EuroPLoP)*, pages 4:1–4:27, Irsee, Germany, 2012. Hillside.

[247] Wenting Ma, Lin Liu, Haihua Xie, Hongyu Zhang, and Jinglei Yin. Preference model driven services selection. In Pascal van Eck, Jaap Gordijn, and Roel Wieringa, editors, *Advanced Information Systems Engineering*, volume 5565 of *Lecture Notes in Computer Science*, pages 216–230. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-02143-5. doi: 10.1007/978-3-642-02144-2_20.

[248] W. Maalej and S. Ghaisas. Capturing and sharing domain knowledge with business rules lessons learned from a global software vendor. In *Requirements Engineering Conference (RE), 2014 IEEE 22nd International*, pages 364–373, Aug 2014. doi: 10.1109/RE.2014.6912287.

[249] Tobias Mahler. *Legal Risk Management*. PhD thesis, University of Oslo, 2010.

[250] N. A. M. Maiden and G. Rugg. Acre: selecting methods for requirements acquisition. *Software Engineering Journal*, 11(3):183–192, May 1996. ISSN 0268-6961.

[251] R. Marler and J. Arora. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26(6):369–395, 2004.

[252] Fabio Massacci, John Mylopoulos, and Nicola Zannone. Security requirements engineering: The si* modeling language and the secure tropos methodology. In Zbigniew Ras and Li-Shiang Tsay, editors, *Advances in Intelligent Information Systems*, volume 265 of *Studies in Computational Intelligence*, pages 147–174. Springer Berlin / Heidelberg, 2010. ISBN 978-3-642-05182-1.

[253] A. K. Massey, R. L. Rutledge, A. I. Anton, and P. P. Swire. Identifying and classifying ambiguity for regulatory requirements. In *Requirements Engineering Conference (RE), 2014 IEEE 22nd International*, pages 83–92, Aug 2014. doi: 10.1109/re.2014.6912250.

[254] Aaron K. Massey, Paul N. Otto, and Annie I. Antón. Aligning requirements with HIPAA in the itrust system. In *16th IEEE International Requirements Engineering Conference, RE 2008, 8-12 September 2008, Barcelona, Catalunya, Spain*, pages 335–336, 2008. doi: 10.1109/re.2008.53.

[255] Aaron K. Massey, Paul N. Otto, and Annie I. Antón. Prioritizing legal requirements. In *Proceedings of the International Workshop on Requirements Engineering and Law (RELAW)*, pages 27–32. IEEE Computer Society, 2009.

[256] Aaron K. Massey, Paul N. Otto, Lauren J. Hayward, and Annie I. Antón. Evaluating existing security and privacy requirements for legal compliance. *Requirements Engineering – Special Issue on Security Requirements Engineering*, 15(1):119–137, 2010.

[257] Aaron K. Massey, Ben Smith, Paul N. Otto, and Annie I. Antón. Assessing the accuracy of legal implementation readiness decisions. In *RE 2011, 19th IEEE International Requirements Engineering Conference, Trento, Italy, August 29 2011 - September 2, 2011*, pages 207–216, 2011. doi: 10.1109/re.2011.6051661.

[258] Jeremy C. Maxwell and Annie I. Antón. Developing production rule models to aid in acquiring requirements from legal texts. In *Proceedings of the 2009 17th IEEE International Requirements Engineering Conference, RE*, RE '09, Washington, DC, USA, 2009. IEEE Computer Society.

[259] Jeremy C. Maxwell and Annie I. Antón. Checking existing requirements for compliance with law using a production rule model. In *Second International Workshop on Requirements Engineering and Law, RELAW 2009, Atlanta, Georgia, USA, September 1, 2009*, pages 1–6, 2009. doi: 10.1109/relaw.2009.3.

[260] Jeremy C. Maxwell, Annie I. Anton, Peter Swire, Maria Riaz, and ChristopherM. McCraw. A legal cross-references taxonomy for reasoning about compliance requirements. *Requirements Engineering*, 17(2):99–115, 2012. ISSN 0947-3602. doi: 10.1007/s00766-012-0152-5.

[261] Jeremy C. Maxwell, Annie I. Antón, and Julie Brande Earp. An empirical investigation of software engineers' ability to classify legal cross-references. In *21st IEEE International Requirements Engineering Conference, RE 2013, Rio de Janeiro-RJ, Brazil, July 15-19, 2013*, pages 24–31, 2013. doi: 10.1109/re.2013.6636702.

[262] Heinrich C. Mayr, Christian Kop, and Daniela Esberger. Business Process Modeling and Requirements Modeling. In *Proceedings of the First International Conference on the Digital Society*, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2760-4. URL `http://portal.acm.org/citation.cfm?id=1260201.1260518`.

[263] Daniel D. McCracken and Edwin D. Reilly. Backus-naur form (bnf). In *Encyclopedia of Computer Science*, pages 129–131. John Wiley and Sons Ltd., Chichester, UK, 2003. ISBN 0-470-86412-5.

[264] J. McDermott and C. Fox. Using abuse case models for security requirements analysis. In *ACSAC '99*, pages 55–64, 1999.

[265] J. E. McGrath. *Groups: Interaction and Performance*. Prentice-Hall, 1984. ISBN 9780133657005. URL `https://books.google.de/books?id=4pzZAAAAMAAJ`.

[266] J. McManus. *Managing Stakeholders in Software Development Projects*. Taylor & Francis, 2007. ISBN 9781136382055. URL `https://books.google.de/books?id=xyyGJ8hb7y8C`.

[267] Gerard Meszaros and Jim Doble. A pattern language for pattern writing. In Robert C. Martin, Dirk Riehle, and Frank Buschmann, editors, *Pattern Languages of Program Design 3*, pages 529–574. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997. ISBN 0-201-31011-2. URL `http://dl.acm.org/citation.cfm?id=273448.273487`.

[268] Hendrik Meth, Alexander Maedche, and Maximilian Einoeder. Is knowledge power? the role of knowledge in automated requirements elicitation. In *Advanced Information Systems Engineering*, volume 7908 of *Lecture Notes in Computer Science*, pages 578–593. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-38708-1. doi: 10.1007/978-3-642-38709-8_37.

[269] Parastoo Mohagheghi, Wasif Gilani, Alin Stefanescu, MiguelA. Fernandez, Björn Nordmoen, and Mathias Fritzsche. Where does model-driven engineering help? experiences from three industrial cases. *Software & Systems Modeling*, 12(3):619–639, 2013. ISSN 1619-1366. doi: 10.1007/s10270-011-0219-7.

[270] N. G. Mohammadi, A. Alebrahim, T. Weyer, M. Heisel, and K. Pohl. A framework for combining problem frames and goal models to support context analysis during requirements engineering. In *CD-ARES '13*, pages 272–288, 2013.

[271] Frank Moisiadis. The fundamentals of prioritising requirements. In *Proceedings of the Systems Engineering, Test and Evaluation Conference (SETE'2002)*, 2002.

[272] C. Monsalve, A. April, and A. Abran. On the expressiveness of business process modeling notations for software requirements elicitation. In *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, pages 3132–3137, Oct 2012. doi: 10.1109/iecon.2012.6389398.

[273] Haralambos Mouratidis and Paolo Giorgini. Secure tropos: a security-oriented extension of the tropos methodology. *International Journal of Software Engineering and Knowledge Engineering*, 17(2):285–309, 2007.

[274] Francisco Moyano, Kristian Beckers, and Carmen Fernandez-Gago. Trust-aware decision-making methodology for cloud sourcing. In Matthias Jarke, John Mylopoulos, Christoph Quix, Colette Rolland, Yannis Manolopoulos, Haralambos Mouratidis, and Jennifer Horkoff, editors, *In the proceedings of the 26th International Conference on Advanced Information Systems Engineering (CAiSE)*, LNCS 8484, pages 136 – 149. Springer, 2014. URL `http://www.springerlink.com/`.

[275] J. Mustajoki and R. Hämäläinen. Smart-swaps — a decision support system for multicriteria decision analysis with the even swaps method. *Decision Support Systems*, 44(1):313 – 325, 2007. ISSN 0167-9236. doi: 10.1016/j.dss.2007.04.004.

[276] J. Mylopoulos, L. Chung, S. Liao, H. Wang, and E. Yu. Exploring alternatives during requirements analysis. *IEEE Software*, 18(1):92–96, 2001.

[277] John Mylopoulos, Lawrence Chung, and Eric Yu. From object-oriented to goal-oriented requirements analysis. *Commun. ACM*, 42(1):31–37, January 1999. ISSN 0001-0782. doi: 10.1145/291469.293165. URL `http://doi.acm.org/10.1145/291469.293165`.

[278] National Institute of Standards and Technology. *Integrated definition methods*, May 1993. `http://www.idef.com/Downloads.htm`.

[279] National Institute of Standards and Technology. Idef1 information modelling, Oktober 2014. http://www.idef.com/pdf/IDEF1MR-part1.pdf, http://www.idef.com/pdf/IDEF1MR-part2.pdf.

[280] National Institute of Standards and Technology. IDEF4 object-oriented design method, Oktober 2014. http://www.idef.com/pdf/Idef4.pdf.

[281] Colin J Neill and Phillip A Laplante. Requirements engineering: the state of the practice. *IEEE software*, (6):40–45, 2003.

[282] Lemai Nguyen and Graeme Shanks. A framework for understanding creativity in requirements engineering. *Inf. Softw. Technol.*, 51(3):655–662, March 2009. ISSN 0950-5849. doi: 10.1016/j.infsof.2008.09.002.

[283] Joaquin Nicolas and Ambrosio Toval. On the generation of requirements specifications from software engineering models: A systematic literature review. *Information and Software Technology*, 51(9):1291 – 1307, 2009. ISSN 0950-5849. doi: 10.1016/j.infsof.2009.04.001. URL `http://www.sciencedirect.com/science/article/pii/S0950584909000378`.

[284] Ali Niknafs and Daniel M. Berry. The impact of domain knowledge on the effectiveness of requirements idea generation during requirements elicitation. In *Requirements Engineering Conference (RE), 2012 20th IEEE International*, pages 181 –190, sept. 2012.

[285] Uolevi Nikula, Jorma Sajaniemi, and Heikki Kälviäinen. *A State-of-the-practice Survey on Requirements Engineering in Small-and Medium-sized Enterprises*. Citeseer, 2000.

[286] J. Noble. Classifying relationships between object-oriented design patterns. In *Proceedings of the Australian Software Engineering Conference*, ASWEC '98, pages 98–107, Washington, DC, USA, 1998. IEEE Computer Society. ISBN 0-8186-9187-5.

[287] Norton. Norton Report 2013. Technical report, Norton, 2013.

[288] David Norton, Mike Blechar, and Teresa Jones. Magic Quadrant for Business Process Analysis Tools. Technical Report February 2010, Gartner, 2010.

[289] Bashar Nuseibeh. Weaving together requirements and architectures. *Computer*, 34(3):115–117, March 2001. ISSN 0018-9162. doi: 10.1109/2.910904.

[290] Bashar Nuseibeh and Steve Easterbrook. Requirements engineering: A roadmap. In *Proceedings of the Conference on The Future of Software Engineering*, ICSE '00, pages 35–46, New York, NY, USA, 2000. ACM. ISBN 1-58113-253-0. doi: 10.1145/336512.336523.

[291] Object Management Group. Business process model and notation (bpmn) version 2.0, jan 2011.

[292] Ray Offen. Domain understanding is the key to successful system development. *Requirements Engineering*, 7(3):172–175, 2002. ISSN 0947-3602. doi: 10.1007/s007660200012.

[293] Ivana Ognjanovic, Dragan Gasevic, Ebrahim Bagheri, and Mohsen Asadi. Conditional preferences in software stakeholders' judgments. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, SAC '11, pages 683–690, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0113-8. doi: 10.1145/1982185.1982335.

[294] OMG. OMG Service oriented architecture Modeling Language, Version 1.0.1. Technical report, Object Management Group, May 2012. URL `http://www.omg.org/spec/SoaML/1.0.1`.

[295] OMG. OMG Systems Modeling Language, Version 1.3, June 2012. URL `http://www.omg.org/spec/SysML/1.3/`.

[296] OMG. OMG Meta Object Facility (MOF) Core Specification, Version 2.4.1, June 2013. URL `http://www.omg.org/spec/MOF/2.4.1`.

[297] OPEN meter project. D1.1 Requirements of AMI. Technical report, OPEN meter project, 2009.

[298] OPEN node project. Evaluation of general requirements according state of the art . Technical report, OPEN node project, 2010.

[299] OPEN node project. Functional Use cases. Technical report, OPEN node project, 2011.

[300] Akira Osada, Daigo Ozawa, Haruhiko Kaiya, and Kenji Kaijiri. The role of domain knowledge representation in requirements elicitation. In *Proceedings of the 25th Conference on IASTED International Multi-Conference: Software Engineering*, SE'07, pages 84–92, Anaheim, CA, USA, 2007. ACTA Press. URL `http://dl.acm.org/citation.cfm?id=1332044.1332059`.

[301] A. F. Osborn. *Applied imagination; principles and procedures of creative problem-solving*. Scribner, 1963. URL `https://books.google.de/books?id=MtZOAAAAMAAJ`.

[302] Paul N. Otto and Annie I. Antón. Addressing legal requirements in requirements engineering. In *Proceedings of the International Conference on Requirements Engineering*. IEEE, 2007.

[303] Carla Pacheco and Ivan Garcia. A systematic literature review of stakeholder identification methods in requirements elicitation. *Journal of Systems and Software*, 85(9):2171 – 2181, 2012. ISSN 0164-1212. doi: 10.1016/j.jss.2012.04.075. URL `http://www.sciencedirect.com/science/article/pii/S0164121212001288`.

[304] Francis Palma, Angelo Susi, and Paolo Tonella. Using an smt solver for interactive requirements prioritization. In *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*, ESEC/FSE '11, pages 48–58, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0443-6. doi: 10.1145/2025113.2025124.

[305] Michael P. Papazoglou, Paolo Traverso, Schahram Dustdar, Frank Leymann, and Bernd J. Krämer. Service-oriented computing: A research roadmap. In Francisco Cubera, Bernd J. Krämer, and Michael P. Papazoglou, editors, *Service Oriented Computing (SOC)*, number 05462 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2006. IBFI, Germany. URL `http://drops.dagstuhl.de/opus/volltexte/2006/524`.

[306] Michael P. Papazoglou, Paolo Traverso, Schahram Dustdar, and Frank Leymann. Service-oriented computing: A research roadmap. In *Int. J. Cooperative Inf. Syst.*, volume 17 of *Int. J. Cooperative Inf. Syst.*, pages 223–255. World Scientific, 2008.

[307] Stefan L. Pauwels, Christian Häbscher, Javier A. Bargas-Avila, and Klaus Opwis. Building an interaction design pattern language: A case study. *Computers in Human Behavior*, 26(3):452 – 463, 2010. ISSN 0747-5632.

[308] Ken Peffers, Tuure Tuunanen, Marcus Rothenberger, and Samir Chatterjee. A design science research methodology for information systems research. *J. Manage. Inf. Syst.*, 24(3):45–77, December 2007. ISSN 0742-1222. doi: 10.2753/mis0742-1222240302.

[309] Mikhail Perepletchikov, Caspar Ryan, Keith Frampton, and Heinz W. Schmidt. Formalising service-oriented design. *Journal of Software*, 3(2), 2008.

[310] Anna Perini, Filippo Ricca, and Angelo Susi. Tool-supported requirements prioritization: Comparing the {AHP} and {CBRank} methods. *Information and Software Technology*, 51(6):1021 – 1032, 2009. ISSN 0950-5849. doi: 10.1016/j.infsof.2008.12.001. URL `http://www.sciencedirect.com/science/article/pii/S0950584908001717`.

[311] Anna Perini, Angelo Susi, and Paolo Avesani. A machine learning approach to software requirements prioritization. *Software Engineering, IEEE Transactions on*, 39(4):445–461, April 2013. ISSN 0098-5589. doi: 10.1109/tse.2012.52.

[312] Kai Petersen, Robert Feldt, Shahid Mujtaba, and Michael Mattsson. Systematic Mapping Studies in Software engineering. In *EASE '08: Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, 2008.

[313] M. Pinto-Albuquerque and A. Rashid. Tackling the requirements jigsaw puzzle. In *Requirements Engineering Conference (RE), 2014 IEEE 22nd International*, pages 233–242, Aug 2014. doi: 10.1109/re.2014.6912265.

[314] Kai Uwe Plath. Datenherausgabepflicht für cloud-anbieter nach us-recht vs. eu-datenschutzrecht. Technical report, CRonline, 2014. URL `http://www.cr-online.de/blog/2014/05/13/datenherausgabepflicht-fuer-cloud-anbieter-nach-us-recht-vs-eu-datenschutzrecht/`.

[315] J.C. Pomerol and S. Barba-Romero. *Multicriterion Decision in Management: Principles and Practice*. Int. series in operations research & management science: ISOR. Kluwer Academic Publishers, 2000. ISBN 9780792377566.

[316] A Pouloudi and EA Whitley. Stakeholder identification in inter-organizational systems: gaining insights for drug use management systems. *European Journal of Information Systems*, 6, 1997.

[317] Rumyana Proynova, Barbara Paech, Andreas Wicht, and Thomas Wetter. Use of personal values in requirements engineering: A research preview. In *Requirements Engineering: Foundation for Software Quality*, volume 6182 of *Lecture Notes in Computer Science*, pages 17–22. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-14191-1. doi: 10.1007/978-3-642-14192-8_3.

[318] Rouzbahan Rashidi-Tabrizi, Gunter Mussbacher, and Daniel Amyot. Transforming regulations into performance models in the context of reasoning for outcome-based compliance. In *Sixth International Workshop on Requirements Engineering and Law, RELAW 2013, 16 July, 2013, Rio de Janeiro, Brasil*, pages 34–43, 2013. doi: 10.1109/relaw.2013.6671344.

[319] Björn Regnell, Barbara Paech, Aybüke Aurum, Claes Wohlin, Allen Dutoit, and Johan Natt Och Dag. Requirements mean decisions! - research issues for understanding and supporting decision-making in requirements engineering. In *in Requirements Engineering, Proc. 1st Swedish Conference on Software Engineering Research and Practice (SERPA'01*, pages 49–52, 2001.

[320] André Rifaut and Sepideh Ghanavati. Measurement-oriented comparison of multiple regulations with GRL. In *Fifth IEEE International Workshop on Requirements Engineering and Law, RELAW 2012, Chicago, IL, USA, September 25, 2012*, pages 7–16, 2012. doi: 10.1109/relaw.2012.6347801.

[321] Linda Rising. *The patterns handbook: Techniques, strategies, and applications*. Cambridge University Press, 1998.

[322] William N. Robinson, Suzanne D. Pawlowski, and Vecheslav Volkov. Requirements interaction management. *ACM Comput. Surv.*, 35:132–190, 2003. ISSN 0360-0300.

[323] Alfonso Rodríguez, Eduardo Fernández-Medina, and Mario Piattini. A bpmn extension for the modeling of security requirements in business processes. *IEICE Transactions*, 90-D(4), 2007.

[324] Guenther Ruhe, Armin Eberlein, and Dietmar Pfahl. Quantitative winwin: A new method for decision support in requirements negotiation. In *Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering*, SEKE '02, pages 159–166, New York, NY, USA, 2002. ACM. ISBN 1-58113-556-4. doi: 10.1145/568760.568789.

[325] Guenther Ruhe, Armin Eberlein, and Dietmar Pfahl. Trade-off analysis for requirements selection. *International Journal of Software Engineering and Knowledge Engineering*, 13(4):345–366, 2003. doi: 10.1142/s0218194003001378.

[326] Per Runeson, Martin Host, Austen Rainer, and Björn Regnell. *Case Study Research in Software Engineering: Guidelines and Examples*. Wiley Blackwell, 2012. ISBN 1118104358.

[327] T. Saaty. The analytic hierarchy and analytic network processes for the measurement of intangible criteria and for decision-making. In J. Figueira, S. Greco, and M. Ehrgott, editors, *Multiple Criteria Decision Analysis: State of the Art Surveys*, pages 345–408. Springer, 2005.

[328] T. Saaty. Decision making with the analytic hierarchy process. *International Journal of Services Sciences*, 1:83–98, 2008.

[329] T. Saaty. The analytic network process. *Iranian Journal of Operations Research*, 1, 2008.

[330] T. Saaty and M. Ozdemir. Negative priorities in the analytic hierarchy process. *Mathematical and Computer Modelling*, 37:1063–1075, 2003.

[331] M. Sadiq and S. K. Jain. Stakeholder identification method in goal oriented requirements elicitation process. In *Requirements Prioritization and Communication (RePriCo), 2014 IEEE 5th International Workshop on*, Aug 2014. doi: 10.1109/reprico.2014.6895219.

[332] E. Sadraei, A. Aurum, G. Beydoun, and B. Paech. A field study of the requirements engineering practice in Australian software industry. *Requirements Engineering*, 2007. URL `http://www.springerlink.com/index/28717Q6Q31N71510.pdf`.

[333] M. Salehie, L. Pasquale, I. Omoronyia, R. Ali, and B. Nuseibeh. Requirements-driven adaptive security: Protecting variable assets at runtime. In *RE '12*, pages 111–120, 2012.

[334] Moshood Omolade Saliu and Guenther Ruhe. Bi-objective release planning for evolving software systems. In *Proceedings of the the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering*, ESEC-FSE '07, pages 105–114, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-811-4. doi: 10.1145/1287624.1287641.

[335] O. Saliu and G Ruhe. Supporting software release planning decisions for evolving systems. In *Software Engineering Workshop, 2005. 29th Annual IEEE/NASA*, pages 14–26, April 2005. doi: 10.1109/sew.2005.42.

[336] SAP. SAP NetWeaver, 2015. URL `http://scn.sap.com/community/netweaver`. last accessed 21.02.2015.

[337] Krishna Sapkota, Arantza Aldea, Muhammad Younas, David A. Duce, and René Bañares-Alcántara. Extracting meaningful entities from regulatory text: Towards automating regulatory compliance. In *RELAW'12*, pages 29–32, 2012.

[338] Christian Schäfer, Thomas Kuhn, and Mario Trapp. A pattern-based approach to dsl development. In *Proceedings of the Compilation of the Co-located Workshops on DSM'11, TMC'11, AGERE! 2011, AOOPES'11, NEAT'11, & VMIL'11*, pages 39–46, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-1183-0. doi: 10.1145/2095050.2095058.

[339] August-Wilhelm W Scheer. *Aris-Business Process Frameworks*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2nd edition, 1998. ISBN 3540644393.

[340] H. Schmidt and J. Jürjens. Connecting security requirements analysis and secure design using patterns and UMLsec. In *CAiSE '11*, pages 367–382. Springer, 2011.

[341] Holger Schmidt. *A Pattern- and Component-Based Method to Develop Secure Software*. PhD thesis, University Duisburg-Essen, April 2010.

[342] Markus Schumacher, Eduardo Fernandez-Buglioni, Duane Hybertson, Frank Buschmann, and Peter Sommerlad. *Security Patterns: Integrating Security and Systems Engineering*. Wiley, 2006.

[343] Till Schümmer and Stephan Lukosch. *Patterns for Computer-Mediated Interaction*. Wiley, 2007.

[344] Peter Schwacke. *Juristische Methodik mit Technik der Fallbearbeitung*. Kohlhammer Deutscher Gemeindeverlag, 4. edition, 2003.

[345] Helen Sharp, Anthony Finkelsteiin, and Galal Galal. Stakeholder identification in the requirements engineering process. In *Proceedings of the 10th International Workshop on Database & Expert Systems Applications*, DEXA '99, pages 387–, Washington, DC, USA, 1999. IEEE Computer Society. ISBN 0-7695-0281-4. URL `http://dl.acm.org/citation.cfm?id=519627.790338`.

[346] Ben Shneiderman. Creating creativity: User interfaces for supporting innovation. *ACM Trans. Comput.-Hum. Interact.*, 7(1):114–138, March 2000. ISSN 1073-0516. doi: 10.1145/344949.345077.

[347] Ben Shneiderman. Creativity support tools: Accelerating discovery and innovation. *Commun. ACM*, 50 (12):20–32, December 2007. ISSN 0001-0782. doi: 10.1145/1323688.1323689.

[348] Siemens. Spectrum Power SOA, 2015. URL `http://w3.siemens.com/smartgrid/global/en/products-systems-solutions/control-center-solutions/grid-control-platform/about-spectrum-power/pages/service-oriented-architecture.aspx`. last accessed 21.02.2015.

[349] A. Siena, A. Perini, and A. Susi. From laws to requirements. In *Proceedings of the International Workshop on Requirements Engineering and Law (RELAW)*, pages 6–10. IEEE, 2008.

[350] A. Siena, A. Perini, A. Susi, and J. Mylopoulos. A meta-model for modelling law-compliant requirements. In *Proceedings of the International Workshop on Requirements Engineering and Law (RELAW)*, pages 45–51. IEEE, 2009.

[351] A. Siena, A. Perini, A. Susi, and J. Mylopoulos. Towards a framework for law-compliant software requirements. In *Software Engineering - Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on*, pages 251–254, May 2009. doi: 10.1109/icse-companion.2009.5070994.

[352] Alberto Siena, John Mylopoulos, Anna Perini, and Angelo Susi. Designing law-compliant software requirements. In AlbertoH. F. Laender, Silvana Castano, Umeshwar Dayal, Fabio Casati, and JosÃ©PalazzoM. de Oliveira, editors, *Conceptual Modeling - ER 2009*, volume 5829 of *Lecture Notes in Computer Science*, pages 472–486. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-04839-5. doi: 10.1007/978-3-642-04840-1_35.

[353] Alberto Siena, Ivan Jureta, Silvia Ingolfo, Angelo Susi, Anna Perini, and John Mylopoulos. Capturing variability of law with nomos 2. In Paolo Atzeni, David Cheung, and Sudha Ram, editors, *Conceptual Modeling*, volume 7532 of *Lecture Notes in Computer Science*, pages 383–396. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-34001-7. doi: 10.1007/978-3-642-34002-4_30.

[354] G. Sindre and A. L. Opdahl. Eliciting security requirements with misuse cases. *Requir. Eng.*, 10(1):34–44, 2005. ISSN 0947-3602.

[355] Sase Narine Singh and Carson Woo. Investigating business-it alignment through multi-disciplinary goal concepts. *Requirements Engineering*, 14(3):177–207, 2009. ISSN 0947-3602. doi: 10.1007/s00766-009-0081-0.

[356] Jim Sinur and Janelle B. Hill. Magic Quadrant for Business Process Management Suites, 2010.

[357] Ghanem Soltana, Elizabeta Fourneret, Morayo Adedjouma, Mehrdad Sabetzadeh, and Lionel C. Briand. Using UML for modeling procedural legal rules: Approach and a study of luxembourg's tax law. In *Model-Driven Engineering Languages and Systems - 17th International Conference, MODELS 2014, Valencia, Spain, September 28 - October 3, 2014. Proceedings*, pages 450–466, 2014. doi: 10.1007/978-3-319-11653-2_28.

[358] I. Sommerville, P. Sawyer, and S. Viller. Viewpoints for requirements elicitation: A practical approach. In *Int. Conf. on RE: Putting Requirements Engineering to Practice*, pages 74–81. IEEE Computer Society, 1998.

[359] Ian Sommerville and Gerald Kotonya. *Requirements Engineering: Processes and Techniques*. John Wiley & Sons, Inc., New York, NY, USA, 1998. ISBN 0471972088.

[360] SpiegelOnline. Globale überwachung: It-firmen erleiden umsatzeinbruch nach snowden-enthüllungen, 12 2013. URL `http://www.spiegel.de/wirtschaft/unternehmen/cisco-und-ibm-erleiden-umsatzeinbruch-nach-nsa-enthuellungen-a-941216.html,Zugriffam08.01.2014`.

[361] Mikkel Stern-Peltz and Jim Armitage. It firms lose billions after nsa scandal exposed by whistleblower edward snowden, 12 2013. URL `http://www.independent.co.uk/life-style/gadgets-and-tech/news/it-firms-lose-billions-after-nsa-scandal-exposed-by-whistleblower-edward-snowden-9028599.html,Zugriffam09.01.2014`.

[362] Ketil Stolen and Ida Solheim. Technology research explained. Technical report, SINTEF, 2007.

[363] A. Sutcliffe and P. Sawyer. Requirements elicitation: Towards the unknown unknowns. In *Requirements Engineering Conference (RE), 2013 21st IEEE International*, pages 92–104, July 2013. doi: 10.1109/re.2013.6636709.

[364] R. B. Svensson, M. Host, and B. Regnell. Managing quality requirements: A systematic review. In *Software Engineering and Advanced Applications (SEAA), 2010 36th EUROMICRO Conference on*, pages 261–268, Sept 2010. doi: 10.1109/seaa.2010.55.

[365] R. B. Svensson, T. Gorschek, B. Regnell, R. Torkar, A. Shahrokni, R. Feldt, and A. Aurum. Prioritization of quality requirements: State of practice in eleven companies. In *Requirements Engineering Conference (RE), 2011 19th IEEE International*, pages 69–78, Aug 2011. doi: 10.1109/re.2011.6051652.

[366] Hirotaka Takeuchi and Ikujiro Nonaka. The new new product development game. *Harvard Business Review*, 1986.

[367] R. Tawhid, E. Braun, N. Cartwright, M. Alhaj, G. Mussbacher, A. Shamsaei, D. Amyot, S. A. Behnam, and G. Richards. Towards outcome-based regulatory compliance in aviation security. In *Requirements Engineering Conference (RE), 2012 20th IEEE International*, pages 267–272, Sept 2012. doi: 10.1109/re.2012.6345813.

[368] Laure-Helene Thevenet and Camille Salinesi. Aligning is to organization's strategy: The instal method. In John Krogstie, AndreasL. Opdahl, and Guttorm Sindre, editors, *Advanced Information Systems Engineering*, volume 4495 of *Lecture Notes in Computer Science*, pages 203–217. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-72987-7. doi: 10.1007/978-3-540-72988-4_15.

[369] Tom Tourwe, Wim Codenie, Nick Boucart, and Vladimir Blagojevic. Demystifying release definition: From requirements prioritization to collaborative value quantification. In Martin Glinz and Patrick Heymans, editors, *Requirements Engineering: Foundation for Software Quality*, volume 5512 of *Lecture Notes in Computer Science*, pages 37–44. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-02049-0. doi: 10.1007/978-3-642-02050-6_4.

[370] Will Tracz. Domain analysis working group report: First international workshop on software reusability. *SIGSOFT Softw. Eng. Notes*, 17(3):27–34, July 1992. ISSN 0163-5948. doi: 10.1145/140938.140940.

[371] I. Trummer, B. Faltings, and W. Binder. Multi-objective quality-driven service selection: A fully polynomial time approximation scheme. *Software Engineering, IEEE Transactions on*, 40(2):167–191, Feb 2014. ISSN 0098-5589. doi: 10.1109/tse.2013.61.

[372] Naoyasu Ubayashi, Toshiki Seto, Hirotoshi Kanagawa, Susumu Taniguchi, Jun Yoshida, Takeshi Sumi, and Masayuki Hirayama. A context analysis method for constructing reliable embedded systems. In *Proceedings of the 2008 International Workshop on Models in Software Engineering*, pages 57–62, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-025-8. doi: 10.1145/1370731.1370745.

[373] T. Ugai and K. Aoyama. Domain knowledge wiki for eliciting requirements. In *Managing Requirements Knowledge (MARK), 2009 Second International Workshop on*, pages 4–6, Sept 2009. doi: 10.1109/mark.2009.4.

[374] UML Revision Task Force. *OMG Unified Modeling Language: Superstructure*, May 2012. `http://www.omg.org/spec/UML/2.4/`.

[375] Unabhängiges Landeszentrum für Datenschutz Schleswig-Holstein. Wirkung von datenschutz- und datensicherheitsaspekten auf wirtschaftliche faktoren, 2014. URL `https://www.datenschutzzentrum.de/projekte/uld-i/tutorial.html;LastAccess:06.10.2014`.

[376] Marjan van den Akker, Sjaak Brinkkemper, Guido Diepen, and Johan Versendaal. Software product release planning through optimization and what-if analysis. *Inf. Softw. Technol.*, 50(1-2):101–111, January 2008. ISSN 0950-5849. doi: 10.1016/j.infsof.2007.10.017.

[377] A. Van Lamsweerde. Elaborating security requirements by construction of intentional anti-models. In *ICSE '04*, pages 148–157, 2004.

[378] Axel van Lamsweerde. *Requirements Engineering: From System Goals to UML Models to Software Specifications*. John Wiley & Sons, 1st edition, 2009.

[379] Axel van Lamsweerde, Emmanual Letier, and Robert Darimont. Managing Conflicts in Goal-Driven Requirements Engineering. *IEEE Trans. Softw. Eng.*, 24(11):908–926, 1998. ISSN 0098-5589. doi: 10.1109/32.730542. URL `http://dx.doi.org/10.1109/32.730542`.

[380] JoseLuis Vara, MichelH. Fortuna, Juan Sánchez, CláudiaM.L. Werner, and MarcosR.S. Borges. A Requirements Engineering Approach for Data Modelling of Process-Aware Information Systems. In Witold Abramowicz, editor, *Business Information Systems SE - 12*, volume 21 of *Lecture Notes in Business Information Processing*, pages 133–144. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-01189-4. doi: 10.1007/978-3-642-01190-0_12. URL `http://dx.doi.org/10.1007/978-3-642-01190-0_12`.

[381] Markus Vehlow and Cordula Golkowsky. *PwC study: Cloud Computing - Navigating the Cloud*. PwC, Frankfurt, October 2010.

[382] Melanie Volkamer. *Evaluation of Electronic Voting: Requirements and Evaluation Procedures to Support Responsible Election Authorities*. Springer Publishing Company, 1st edition, 2009. ISBN 3642016618, 9783642016615.

[383] Melanie Volkamer and Roland Vogt. Common Criteria Protection Profile for Basic set of security requirements for Online Voting Products. Technical report, Bundesamt f'ur Sicherheit in der Informationstechnik, 2008. URL `https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/ReportePP/pp0037b_engl_pdf.pdf?__blob=publicationFile`.

[384] Graham Wallas. *The art of Thought*. Harcourt, 1926.

[385] A. Webb. Triz: an inventive approach to invention. *Engineering Management Journal*, 12(3):117–124, June 2002. ISSN 0960-7919. doi: 10.1049/em:20020302.

[386] Martijn Van Welie and Gerrit C. Van Der Veer. Pattern languages in interaction design: Structure and organization. In *Proc. Interact '03, M. Rauterberg, Wesson, Ed(s). IOS*, pages 527–534. IOS Press, 2003.

[387] Karl E Wiegers. Karl wiegers describes 10 requirements traps to avoid. 2000.

[388] Roel Wieringa. *Design Science Methodology for Information Systems and Software Engineering*. Springer, 2014. ISBN 978-3-662-43838-1. doi: 10.1007/978-3-662-43839-8.

[389] Roel Wieringa and Ayşe Moralı. Technical action research as a validation method in information systems design science. In *Proceedings of the 7th International Conference on Design Science Research in Information Systems: Advances in Theory and Practice*, DESRIST'12, pages 220–238, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 978-3-642-29862-2. doi: 10.1007/978-3-642-29863-9_17.

[390] Jennifer Wiley. Expertise as mental set: The effects of domain knowledge in creative problem solving. *Memory & Cognition*, 26(4):716–730, 1998. ISSN 0090-502X. doi: 10.3758/bf03211392.

[391] R. R. Willis. *Hughes Aircraft's Widespread Deployment of a Continuously Improving Software Process*. AD-a358 993. Carnegie-mellon university, 1998.

[392] Tiffany Winn and Paul Calder. A pattern language for pattern language structure. In *Proceedings of the 2002 Conference on Pattern Languages of Programs - Volume 13*, CRPIT '02, pages 45–58, Darlinghurst, Australia, Australia, 2003. Australian Computer Society, Inc. ISBN 0-909-92591-7. URL `http://dl.acm.org/citation.cfm?id=1151669.1151673`.

[393] Claes Wohlin, Per Runeson, Martin Host, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in Software Engineering: An Introduction*. Kluwer Academic Publishers, Norwell, MA, USA, 2014. ISBN 0-7923-8682-5.

[394] Chao-Tung Yang, Chung-Che Lai, Wei-Sheng Chen, Jung-Chun Liu, and WilliamC. Chu. Implementation of a smart grid system with soa-based service on cloud. In Ruay-Shiung Chang, Tai-hoon Kim, and Sheng-Lung Peng, editors, *Security-Enriched Urban Computing and Smart Grid*, volume 223 of *Communications in Computer and Information Science*, pages 159–168. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-23947-2. doi: 10.1007/978-3-642-23948-9_19.

[395] R. K. Yin. *Case Study Research: Design and Methods*. Applied Social Research Methods. SAGE Publications, 2009. ISBN 9781412960991. URL `https://books.google.de/books?id=FzawIAdilHkC`.

[396] E. Yu. *Modelling strategic relationships for process reengineering*. PhD thesis, University of Toronto, 1996.

[397] E. Yu. Towards modeling and reasoning support for early-phase requirements engineering. In *Proc. of the 3rd IEEE Int. Symp. on Requirements Engineering*, RE '97, pages 226–. IEEE Computer Society, 1997. ISBN 0-8186-7740-6.

[398] Pamela Zave. Classification of research efforts in requirements engineering. *ACM Comput. Surv.*, 29(4): 315–321, December 1997. ISSN 0360-0300. doi: 10.1145/267580.267581.

[399] Pamela Zave and Michael Jackson. Four dark corners of requirements engineering. *ACM Trans. Softw. Eng. Methodol.*, 6:1–30, 1997. ISSN 1049-331X.

[400] Uwe Zdun. Systematic pattern selection using pattern language grammars and design space analysis. *Softw. Pract. Exper.*, 37(9):983–1016, July 2007. ISSN 0038-0644. doi: 10.1002/spe.v37:9.

[401] Kiyavitskaya Zeni, Nicola. Gaiust: supporting the extraction of rights and obligations for regulatory compliance. *Requirements Engineering*, 20(1):1–22, 2015. doi: 10.1007/s00766-013-0181-8.

[402] Reinhold Zippelius. *Juristische Methodenlehre*. Beck, 2012.

[403] Werner Zwick. Standardisierung im datenschutz: Auswirkungen in der praxis. *Datenschutz und Datensicherheit- DuD*, (1), 2006.